НИИ «НАУЧНЫЙ ЦЕНТР»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДВК

КНИГА 4

**ЯЗЫКИ ПРОГРАММИРОВАНИЯ:
ФОРТРАН, МОДУЛА-2**

НИИ «НАУЧНЫЙ ЦЕНТР»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДВК

КНИГА 4

ЯЗЫКИ ПРОГРАММИРОВАНИЯ ФОРТРАН, МОДУЛА-2

МОСКВА 1990 год

АННОТАЦИЯ

В данную книгу включены сведения о системе ФОРТРАН ОС ФОДОС-2 и о языке МОДУЛА-2.

В главе «ОПИСАНИЕ ПРИМЕНЕНИЯ» описывается назначение системы ФОРТРАН, ее состав и общие положения по системе.

Глава «ОПИСАНИЕ ЯЗЫКА» содержит сведения об элементах языка, элементах данных, выражениях, операторах, средствах обмена данными, объявлениях, процедурах и модулях, а также о средствах отладки программ.

Глава «ТРАНСЛЯТОР С ФОРТРАНа» включает в себя описание основных характеристик и особенностей транслятора, процедур вызова транслятора, структуры входных и выходных данных, приведены сообщения об ошибках и действиях, которые необходимо предпринять по этим сообщениям.

Глава «БИБЛИОТЕКА ФОРТРАНа» является руководством по использованию библиотеки ФОРТРАНа и содержит общие сведения о библиотеке, описание характеристик и особенностей библиотеки ФОРТРАНа, сведения об условиях, необходимых для вызова подпрограмм библиотеки.

В главе «ДИАЛОГОВЫЙ ОТЛАДЧИК» приведены общие сведения об этой программе, даны ее характеристики и описаны входные и выходные данные.

Глава «КОНТРОЛЬНЫЕ ЗАДАЧИ» представляет собой руководство по работе с программами DEMO.FOR и DEMO1.FOR, которые используются в качестве контрольных задач для проверки на работоспособность транслятора и библиотеки ФОРТРАНа.

Глава «СИСТЕМА УПРАВЛЕНИЯ ГРАФИЧЕСКИМ ТЕРМИНАЛОМ. ГРАФИЧЕСКИЙ ПАКЕТ» содержит:

— обзор графического пакета;
— характеристики графического пакета;
— обращения к подпрограммам пакета;
— описания графических подпрограмм;
— описание общей области состояния терминала;
— инструкции по работе с библиотекой графических подпрограмм;
— словарь основных терминов системы управления терминалом.

Две последние главы включают в себя описание языка МОДУЛА-2, созданного профессором Н. Виртом и информацию, необходимую программисту для работы с языком МОДУЛА-2 в ОС ФОДОС-2. Описание языка дано в сжатом виде.

ФОРТРАН/ФОДОС-2. ОПИСАНИЕ ПРИМЕНЕНИЯ

1. НАЗНАЧЕНИЕ СИСТЕМЫ

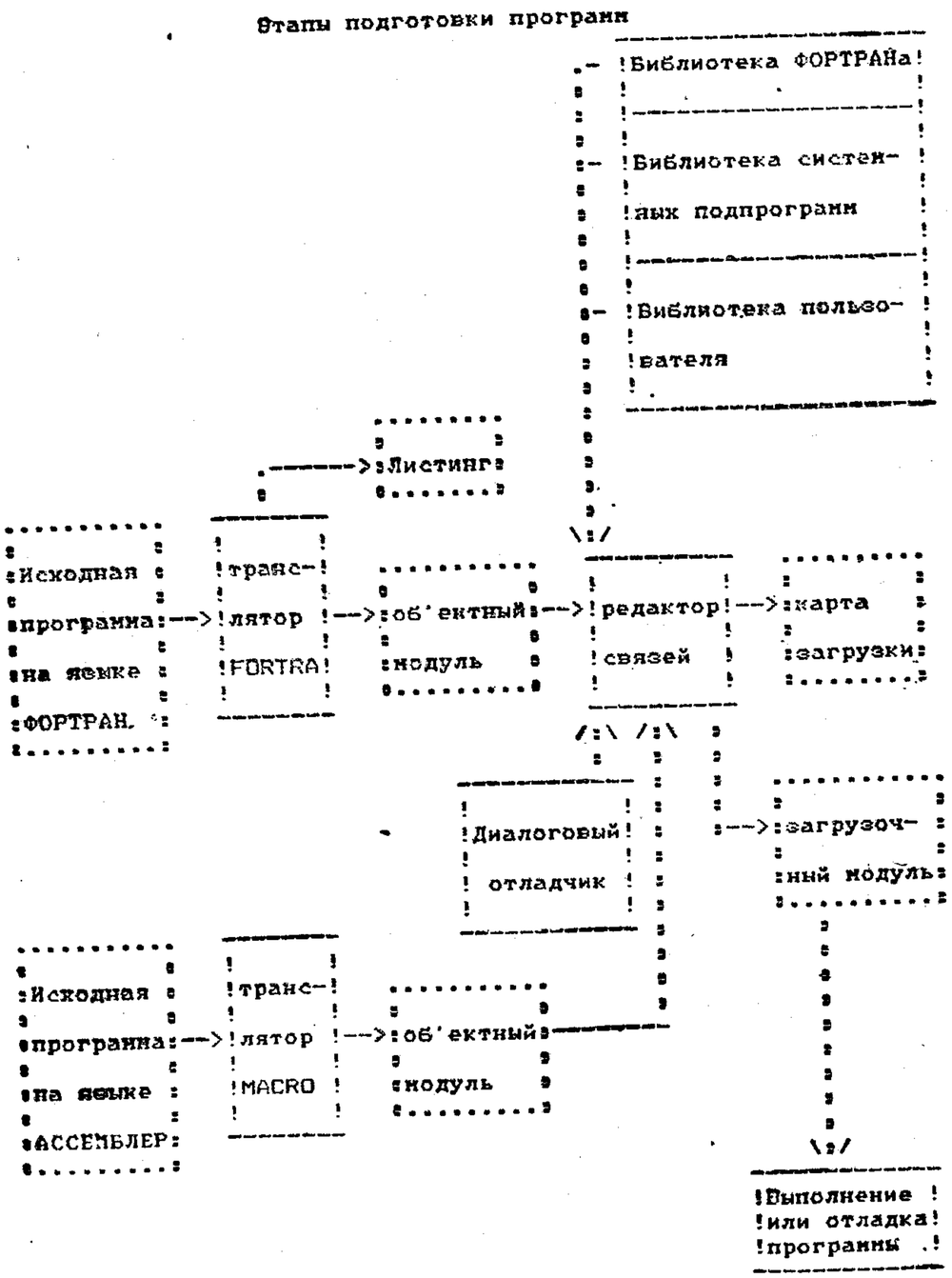
Система ФОРТРАН предназначена для автоматизации программирования на уровне проблемно-ориентированного языка ФОРТРАН. Она работает под управлением операционной системы ФОДОС-2 [1] и применяется для решения различного рода научно-технических и инженерных задач.

Система ФОРТРАН представляет пользователю следующие возможности:

— разработку программ на языке ФОРТРАН;
— использование средств операционной системы ФОДОС-2;
— использование внешних функций языка ФОРТРАН;
— обращение к подпрограммам, написанным на языке АССЕМБЛЕР;
— получение объектного модуля, пригодного для дальнейшей обработки его программами операционной системы ФОДОС-2;
— получение листинга программы и выдачу информации о распределении памяти;
— диагностику ошибок на этапе трансляции и выполнения программ;
— отладку программ, написанных на языке ФОРТРАН.

Подготовка и выполнение программ, написанных на языке ФОРТРАН, осуществляется под управлением операционной системы ФОДОС-2. Этапы подготовки программ приведены на рисунке, они содержат:

— получение исходного файла (стандартный тип .FOR) с помощью программы редактор текста;
— трансляцию исходного файла и получение объектного модуля и листинга программы;
— связывание объектного модуля, полученного после трансляции, с подпрограммами библиотеки ФОРТРАНа и библиотеки системных подпрограмм;
— если есть необходимость, связывание объектного модуля,



полученного после трансляции, с файлом диалогового отладчика или библиотекой пользователя;
— получение загрузочного модуля и карты загрузки;
— загрузка и запуск программы по команде монитора R или RUN.

Программы, написанные на языке ФОРТРАН, можно связывать с программами, написанными на языке АССЕМБЛЕР, в единый загрузочный модуль с помощью программы редактор связей.

2. УСЛОВИЯ ПРИМЕНЕНИЯ

2.1. Требования к техническим средствам

Для функционирования системы ФОРТРАН требуется следующая минимальная конфигурация технических средств:
— процессор;
— оперативное запоминающее устройство емкостью 16К слов для работы под управлением мониторов SJ и FB; 32К слов и диспетчер памяти для работы под управлением монитора XM;
— терминал;
— устройство ввода-вывода информации на гибких магнитных дисках или накопитель на жестком диске.

Система ФОРТРАН обеспечивает обслуживание следующих устройств:
— магнитной ленты;
— кассетной магнитной ленты;
— построчно-печатающих устройств типа robotron СМ 6329.02;
— перфоленточных устройств ввода-вывода.

2.2. Требования к программным средствам

Для использования системы ФОРТРАН необходимы следующие программные средства:

компоненты операционной системы ФОДОС-2:
FMON<YY>.SYS — монитор системы;
SWAP.SYS — файл свопинга;
<XX>.SYS — драйвер диска;
TT.SYS — драйвер терминала;
PIP.SAV — программа обмена;
DUP.SAV — программа обслуживания устройств;
DIR.SAV — программа получения справочника;
LINK.SAV — редактор связей;
LIBR.SAV — библиотекарь;
K13.SAV — редактор текста;

SYSLIB.OBJ — библиотека системных подпрограмм.
где — YY — SJ, FB или XM;
XX — DX, DY, MX, MY, DW.

3. ОПИСАНИЕ СИСТЕМЫ

В этом разделе описаны назначение и функции основных компонентов рабочей системы ФОРТРАН:
FORTRA.SAV — транслятор с ФОРТРАНа;
FORLIB.OBJ — библиотека ФОРТРАНа;
UIOBYT.OBJ — файл бесформатного ввода-вывода;
FDT.OBJ — диалоговый отладчик;
DEMO.FOR, DEMO1.FOR — контрольные задачи;
FORTRA.HLP — вспомогательный файл.

3.1. Транслятор с ФОРТРАНа

Транслятор с ФОРТРАНа (файл FORTRA.SAV) предназначен для преобразования исходной программы, написанной на языке ФОРТРАН, в эквивалентную ей объектную программу.

Транслятор осуществляет следующие функции:
— управление процессом трансляции с помощью переключателей командной строки;
— распознавание предложений языка ФОРТРАН;
— диагностический контроль и выдачу сообщений об ошибках;
— получение объектного модуля, удовлетворяющего требованиям операционной системы ФОДОС-2;
— получение листинга программы с информацией о распределении памяти;
— определение символических имен подпрограмм библиотеки ФОРТРАНа, которые требуются для выполнения программы пользователя.

Режим работы транслятора задается введением с терминала командной строки. Подробное описание транслятора и работа с ним приведены в документе [2].

3.2. Библиотека ФОРТРАНа

Библиотека ФОРТРАНа (файл FORLIB.OBJ) содержит набор объектных модулей, которые обеспечивают выполнение:
— операторов языка ФОРТРАН;
— внешних функций языка ФОРТРАН;

— операций ввода-вывода;
— вспомогательных подпрограмм;
— различных смешанных функций.

Список внешних функций приведен в приложении 1, список вспомогательных подпрограмм библиотеки ФОРТРАНа приведен в табл. 1.

Кроме того, библиотека обеспечивает обработку:
— многомерных массивов;
— виртуальных массивов;
— ошибок различных видов.

Необходимые модули библиотеки ФОРТРАНа включаются в загрузочный модуль программы пользователя с помощью редактора связей. Подробное описание библиотеки ФОРТРАНа приведено в документе [3].

ВСПОМОГАТЕЛЬНЫЕ ПОДПРОГРАММЫ БИБЛИОТЕКИ ФОРТРАНа

Таблица 1

Имя подпрограммы	Назначение подпрограммы
1	2
ASSIGN	Устанавливает соответствие между спецификацией файла и номером логического устройства
CLOSE	Закрывает файл на указанном логическом устройстве
DATE	Определяет текущую дату в виде последовательности символов
IDATE	Определяет текущую дату в виде трех целых чисел
EXIT	Заканчивает выполнение программы пользователя и передает управление монитору
USEREX	Задает имя подпрограммы завершения
RANDU	Генерирует случайное число на интервале от 0 до 1
SETERR	Изменяет реакцию библиотеки ФОРТРАНа по ошибке
ERRTST	Контролирует появление ошибки, указанной пользователем
ERRSNS	Регистрирует последнюю обнаруженную ошибку

3.3. Файл бесформатного ввода-вывода (UIOBYT.OBJ)

обеспечивает бесформатный ввод-вывод для чтения и записи данных по байтам.

Этот файл, при необходимости, может включаться в библиотеку системных подпрограмм или библиотеку ФОРТРАНа

с помощью программы библиотекарь. Формат командной строки приведен ниже:

— для включения в SYSLIB.OBJ

```
.R LIBR
*уст:SYSLIB[—1]=уст:SYSLIB,уст:UIOBYT/U/G
GLOBAL? $ERRS <BK>
GLOBAL? $ERRTB <BK>
GLOBAL? $OVRH <BK>
GLOBAL? <BK>
СУ/С
```

— для включения в FORLIB.OBJ

```
.R LIBR
*уст:FORLIB[—1]=уст:FORLIB,уст:UIOBYT/U/G
GLOBAL? $ERRS <BK>
GLOBAL? $ERRTB <BK>
GLOBAL? <BK>
СУ/С
```

— для включения в FORLIB.JFB

```
.R LIBR
*уст:FORLIB.JFB[—1]=уст:FORLIB.JFB,уст:UIOBYT/U/G
GLOBAL? $ERRS <BK>
GLOBAL? $ERRTB <BK>
GLOBAL? <BK>
СУ/С
```

где уст — устройство, на котором находится файл SYSLIB .OBJ, FORLIB.OBJ, FORLIB.JFB или UIOBYT.OBJ.

3.4. Диалоговый отладчик (файл FDT.OBJ) предназначен для проведения отладки программ, написанных на языке ФОРТРАН.

3.5. Контрольные задачи (DEMO.FOR, DEMO1.FOR) используются для проверки на работоспособность транслятора и библиотеки ФОРТРАНа под управлением любого из мониторов (XM, SJ, FB) системы ФОДОС-2.

3.6. Вспомогательный файл (FORTRA.HLP) содержит описание переключателей транслятора с ФОРТРАНа. Вызов вспомогательного файла осуществляется после запуска транслятора. Текст вспомогательного файла приведен в приложении 2.

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Входными данными для системы ФОРТРАН являются файлы исходных программ, написанных на языке ФОРТРАН.

Результатом работы транслятора являются два выходных файла: объектный файл и файл листинга.

Объектный файл содержит программу в одном из двух объектных кодов: нанизанном или поточном.

Нанизанному коду соответствует объектный код программы в виде обращения к подпрограммам библиотеки ФОРТРАНа. Поточному коду соответствует объектный код программы на языке АССЕМБЛЕРа.

Листинг выдается транслятором по полям. Основными полями листинга являются:

— поле исходной программы;
— поле диагностики;
— поле карты памяти;
— поле генерированного кода.

ПРИЛОЖЕНИЕ 1

ВНЕШНИЕ ФУНКЦИИ

В таблице данного приложения приведены внешние функции языка ФОРТРАН операционной системы ФОДОС-2.

Таблица

Функция	Символическое имя	Тип параметров	Тип функции
1	2	3	
Абсолютное значение	ABS	вещ.	вещ.
	IABS	цел.	цел.
	DABS	дв.	дв.
	CABS	компл.	вещ.
Усечение	AINT	вещ.	вещ.
	INT	вещ.	цел.
	IDINT	дв.	цел.

1	2	3	
Взятие остатка	AMOD	вещ.	вещ.
	MOD	цел.	цел.
	DMOD	дв.	дв.
Выбор наибольшего значения	AMAX0 AMAX1	цел. вещ.	вещ. вещ.
	MAX0	цел.	цел.
	MAX1	вещ.	цел.
	DMAX1	дв.	дв.
Выбор наименьшего значения	AMIN0	цел.	вещ.
	AMIN1	вещ.	вещ.
	MIN0	цел.	цел.
	MIN1	вещ.	цел.
	DMIN1	дв.	дв.
Преобразование в плавающую форму	FLOAT	цел.	вещ.
Преобразование в фиксированную форму	IFIX	вещ.	цел.
Передача знака	SIGN	вещ.	вещ.
	ISIGN	цел.	цел.
	DSIGN	дв.	дв.

1	2	3	
Положительная разность	DIM IDIM	вещ. цел.	вещ. цел.
Получение максимальной значащей части параметра двойной точности	SNGL	дв.	вещ.
Получение действительной части комплексного параметра	REAL	компл.	вещ.
Получение мнимой части комплексного параметра	AIMAG	компл.	вещ.
Преобразование вещественного параметра в форму двойной точности	DBLE	вещ.	дв.
Преобразование двух вещественных параметров в комплексную форму	CMPLX	вещ.	компл.
Получение комплексной величины, сопряженной с параметром	CONJG	компл.	компл.
Экспоненциальная функция	EXP	вещ.	вещ.
	DEXP	дв.	дв.
	CEXP	компл.	компл.
Натуральный логарифм	ALOG	вещ.	вещ.
	DLOG	дв.	дв.
	CLOG	компл.	компл.
Десятичный логарифм	ALOG10	вещ.	вещ.
	DLOG10	дв.	дв.

1	2	3	
Тригонометрический синус	SIN	вещ.	вещ.
	DSIN	дв.	дв.
	CSIN	компл.	компл.
Тригонометрический косинус	COS	вещ.	вещ.
	DCOS	дв.	дв.
	CCOS	компл.	компл.
Гиперболический тангенс	TANH	вещ.	вещ.
Квадратный корень	SQRT	вещ.	вещ.
	DSQRT	дв.	дв.
	CSQRT	компл.	компл.
Арктангенс	ATAN	вещ.	вещ.
	DATAN	дв.	дв.
	ATAN2	вещ.	вещ.
	DATAN2	дв.	дв.
Генератор случайного числа	RAN	цел.	вещ.

Сокращения: цел. — целый, вещ. — вещественный, компл. — комплексный, дв. — двоичный.

ТЕКСТ ФАЙЛА FORTRA.HLP

Ниже приведен текст вспомогательного файла FORTRA.HLP.

/ALLOCATE:M	—	резервируем M блоков памяти на томе для объектного файла и файла листинга. Задается после переключателя /OBJECT или /LIST
/CODE:XXX	/I:XXX	определяет вид объектного кода: — поточный с использованием команд расширенной арифметики (XXX=EIS); — поточный с использованием команд расширенной арифметики и плавающей запятой (XXX=FIS); — нанизанный (XXX=THR)
/DIAGNOSE	/B	разрешает вывод расширения поля генерированного (поточного) кода листинга
/EXTEND	/E	разрешает расширение строк исходной программы до 80 символов
/HEADER	/O	разрешает вывод поля режима работы транслятора
/I4	/T	резервирует два слова памяти для целых переменных, не имеющих явного указания длины. Значение целой переменной размещается в слове с младшим адресом. По умолчанию резервируется одно слово памяти
/LINENUMBERS	—	разрешает вывод номеров внутренней последовательности в объектный код
/LIST[:спф]	—	вывод листинга
/NOLINENUMBERS	/S	запрещает вывод номеров внутренней последовательности в объектный код

/NOOBJECT	—	запрещает вывод объектного файла
/NOSWAP	/U	запрещает свопинг USR во время работы программы пользователя
/NOVECTORS	/V	запрещает векторизацию многомерных массивов
/OBJECT[:спф]	—	вывод объектного файла
/ONDEBUG	/D	разрешает трансляцию строк отладки. По умолчанию строки отладки рассматриваются как комментарии
/RECORD:M	/R:M	определяет максимальный размер записи в байтах для операторов форматного ввода-вывода (4<=M<==4095). По умолчанию M=136 байт
/SHOW[:M]	/L[:M]	управляет основными полями листинга (0<=M<=7) 0 — печать сообщений об ошибках 1 или SRC — печать исходной программы и сообщений об ошибках 2 или MAP — печать карты памяти и сообщений об ошибках 4 или COD — печать генерированного кода и сообщений об ошибках 7 или ALL — печать основных полей листинга

/STATISTICS	/A	разрешает вывод поля статистики листинга
/SWAP	—	разрешает свопинг USR во время работы программы пользователя
/UNITS:M	/N:M	определяет максимальное число логических устройств (1<=M<=15), которые могут быть открыты одновременно. По умолчанию M=6
/VECTORS	—	разрешает векторизацию многомерных массивов
/WARNINGS	/W	разрешает вывод сообщений, предупреждающих об ошибках. Используется с переключателем /SHOW или /L
	/H	вывод файла FORTRA.HLP на терминал
	/Q	запрещает в листинге вывод заголовка программного модуля (головного или FUNCTION, SUBROUTINE и BLOCK DATA)
	/X	зарезервирован для расширения возможностей транслятора
	/Z	присваивает программным секциям $CODE и $DATAP значение признака доступа R0

ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Операционная система ФОДОС-2
   Введение в программное обеспечение
   Описание применения
2. ФОРТРАН/ФОДОС-2
   Транслятор с ФОРТРАНа
   Руководство программиста
3. ФОРТРАН/ФОДОС-2
   Библиотека ФОРТРАНа
   Руководство программиста

ФОРТРАН/ФОДОС-2. ОПИСАНИЕ ЯЗЫКА.

1. ОБЩИЕ СВЕДЕНИЯ

Язык ФОРТРАН операционной системы ФОДОС-2 полностью согласуется со стандартом языка ФОРТРАН и, кроме того, содержит новые средства:

1) по элементам и правилам записи программ:
— символы алфавита — ', ", !, :,
— строки отладки;
2) по элементам данных:
— тип байтовый,
— восьмеричные константы,
— константы RADIX-50,
— буквенно-цифровые литералы;
3) по выражениям:
— выражения смешанного типа,
— логические операции — .XOR., .EQV.;
4) по операторам ввода-вывода:
— параметры END=S и ERR=S в операторах ввода-вывода,
— операторы форматного ввода-вывода ACCEPT, TYPE, PRINT,
— операторы с преобразованием по списку,
— операторы прямого доступа,
— вспомогательные операторы OPEN,CLOSE,DEFINE,FILE, FIND,
— операторы передачи данных ENCODE и DECODE,
— первый символ $ при выводе форматной записи на печать;
5) по объявлениям:
— объявления VIRTUAL, IMPLICIT,
— заголовок головного модуля PROGRAM,
— описатели полей ROW, '...', TN, Q, :, $ в спецификации формата;
6) по функциям:
— RAN.

Язык ФОРТРАН операционной системы ФОДОС-2 расширяет некоторые положения стандартного языка ФОРТРАН:

1) по правилам записи программ:
— запись комментариев в предложении;
2) по описанию массивов:
— число элементов массива до семи,
— представление индексного выражения в виде любого допустимого арифметического выражения;
3) по операторам управления:
— в операторе GOTO (K1, K2, ..., KN),I I — арифметическое выражение,
— в операторе DO параметры M1, M2, M3 — арифметические выражения, M3 может принимать отрицательное значение, закрывающим оператором может быть оператор PAUSE,
— в операторах STOP N и PAUSE NN — последовательность десятичных или восьмеричных цифр или буквенно-цифровой литерал;
4) по операторам ввода-вывода:
— элементы списка вывода в виде констант и выражений,
— запятая для разграничения внешних полей ввода,
5) по объявлениям:
— описатели полей в спецификации формата в виде кодов преобразования,
— указатели длины в объявлении типа и заголовка функции,
— символическое имя в заголовке BLOCK DATA.

2. СПОСОБ ОПИСАНИЯ ЯЗЫКА

В документе используются следующие соглашения:
— элемент или группа элементов, заключенные в квадратные скобки ([ ]), являются необязательными;
— элемент или группа элементов, предшествующие многоточию (...), могут быть повторены;
— элементы из латинских букв, являющиеся ключевыми словами (в форматах предложений они подчеркнуты), записываются точно так же, как они записаны в формате предложения;
элементы, следующие за ключевым словом, заменяются согласно описанию;
сочетание символов «<=» обозначает операцию «меньше или равно»;
сочетание символов «>=» обозначает операцию «больше или равно»;
символ «_» (подчеркивание) указывает на пробел.

3. ЭЛЕМЕНТЫ ЯЗЫКА

3.1. Структура языка

Программа, записанная на языке ФОРТРАН, состоит из одного или нескольких программных модулей и содержит информацию о форме записи исходных данных, алгоритмах их переработки, а также о вводимых внутренних объектах и форме представления окончательных результатов.

Один из программных модулей является головным. Выполнение программы начинается с выполнения ее головного модуля.

Каждый программный модуль состоит из предложений и комментариев. В этом смысле головной модуль — это последовательность предложений и комментариев языка ФОРТРАН, не содержащая заголовков функций, заголовков подпрограмм и заголовков спецификаций блоков данных. Модуль, не являющийся головным, начинается с одного из перечисленных выше заголовков.

В любом программном модуле (в том числе и головном), за исключением модуля — блока данных, могут использоваться внешние процедуры (разд. 9), описывающие отдельные процедуры процесса обработки данных вне данного модуля.

Предложения языка ФОРТРАН распадаются на два класса: выполняемые (или операторы) и невыполняемые (или объявления). Операторы определяют действия в программе, тогда как объявления описывают способ использования программы, характеристики операндов, способ редактирования данных, вводимые в употребление функции или размещение данных.

Предложение делится на физические части, называемые строками (п. 3.3), первая из которых называется начальной строкой, а остальные — строками-продолжениями.

Комментарий — это строка или часть строки, которая не является предложением или его частью.

Синтаксическими элементами предложения являются имена и операции. Имена используются для ссылок на объекты. Операции определяют действия над именованными объектами.

Один из частных случаев имен — имя массива. С именем массива должен быть связан размер идентифицируемого массива, определяемый в описании массива (п. 8.1.1). Имя массива, дополненное индексом, используется для идентификации конкретного элемента массива (п. 4.3.3.1).

Имена данных и операции могут быть связаны в выражения, которые служат для задания правил вычисления некоторого значения. Это значение получается в результате выполнения указанных в выражении операций над именованными данными.

3.2. Алфавит языка. При записи программного модуля используются символы, входящие в алфавит языка ФОРТРАН. Алфавит делится на три группы символов: цифры, буквы и специальные символы.

3.2.1. Цифры. Десятичная цифра — это один из десяти символов:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Восьмеричная цифра — это один из восьми символов:
0; 1, 2, 3, 4, 5, 6, 7.

В языке ФОРТРАН используются как десятичные, так и восьмеричные числа.

Цифра или последовательность цифр, если не оговорено, интерпретируется как десятичная цифра или десятичное число.

3.2.2. Буквы. Буква — это один из двадцати шести символов:
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.

3.2.3. Буквенно-цифровые символы — это либо буква, либо цифра.

3.2.4. Специальные символы — это один из символов, приведенных в табл. 1.

Таблица 1

Символ	Название символа
1	2
	пробел
=	равно
+	плюс
—	минус
*	звездочка
/	дробная черта
(	круглая левая скобка (левая скобка)
)	круглая правая скобка (правая скобка)
,	запятая
.	точка
'	апостроф
"	кавычки
:	двоеточие
$	знак денежной единицы
!	восклицательный знак

Символ пробела — это отсутствие какого-либо графического изображения в данной позиции. Кроме специально оговоренных случаев (п.п. 3.3.2—3.3.4, 4.2.6, 4.3.1.7, 7.2.1), символ пробела не является значащим, и поэтому используется для улучшения наглядности программ.

Дополнительно к перечисленным в табл. 1 символам может быть использован символ ГТ (горизонтальная табуляция) для перемещения на восемь позиций печати. Символ ГТ записывается только в начале строки.

3.3. Строки — это последовательность, состоящая из 72 символов с возможностью расширения до 80 символов. Каждый символ должен принадлежать алфавиту ФОРТРАН, за исключением случаев, описанных в п.п. 4.2.6, 4.3.1.7, 7.2.1. Позиции символов в строке нумеруются слева направо от 1 до 80 включительно.

Позиции 1—5 служат для записи меток предложений. Кроме того, в позиции 1 может указываться буква «С» или «D». Буква «С» определяет строку как комментарий, букв «D» — как строку отладки.

Позиция 6 служит для записи признака продолжения, позиции 7—72 — для записи предложений, позиции 73—80 — для идентификации программы, номера следования и т. д.

Строка, состоящая из пробелов или не содержащая никаких символов, является пустой строкой. Пустая строка игнорируется.

3.3.1. Комментарии. Буква «С» в позиции 1 указывает на то, что данная строка является комментарием. За строкой-комментарием должны непосредственно следовать либо другая строка-комментарий, либо начальная строка, либо заключительная строка.

Комментарий может содержаться также в любой строке предложения. Символ «!», за исключением случаев, описанных в п.п. 4.2.6, 4.3.1.7, 7.2.1, указывает на то, что следующая за ним последовательность символов является комментарием.

Комментарий не оказывает никакого влияния на выполнение программы, его используют для пояснений.

3.3.2. Строка отладки. Буква «D» в позиции 1 указывает на то, что данная строка является строкой отладки. Строка отладки не может содержать метку в позициях 2—5. За строкой отладки могут следовать строки продолжения.

Строки отладки рассматриваются как комментарии до тех пор, пока не будет задан соответствующий переключатель (/ONDEBUG или /D) в командной строке (см. [2]).

3.3.3. Заключительная строка — это строка, содержащая

в позициях 1—6 пробелы, а в позициях 7—72 — пробелы и буквы E, N и D. Эти буквы должны следовать в том порядке, в каком они приведены, и могут размещаться в любой из этих позиций, в остальных позициях должны содержаться пробелы. Заключительная строка указывает на конец текста программного модуля. Текст каждого программного модуля обязательно должен завершаться одной заключительной строкой.

3.3.4. Начальная строка — это такая строка, которая не является ни комментарием, ни заключительной строкой и содержит пробел или цифру 0 в позиции 6.

В позициях 1—5 содержится либо метка предложения, либо пробелы.

3.3.5. Строка-продолжение — это такая строка, которая не является комментарием и в позиции 6 содержит символ, отличный от пробела и цифры 0. Строка-продолжение может непосредственно следовать только за начальной строкой или за другой строкой-продолжением.

Символы, начиная с позиции 7 строки-продолжения, рассматриваются как продолжение предыдущей строки.

3.4. Предложения. Предложение состоит из одной начальной строки, за которой могут следовать строки-продолжения. Символы, образующие предложение, записываются в позициях 7—72 каждой из строк и являются упорядоченными: сначала идут символы, записанные в начальной строке, затем — символы, записанные в первой строке-продолжении (если она имеется) и т. д.

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

3.6. Символические имена. Символическое имя состоит из буквы, за которой могут следовать от одного до пяти буквенно-цифровых символов. Ниже приведены примеры правильных и неправильных записей символических имен.

Правильная запись	Неправильная запись
NUMBER	5Q (начинается с цифры)

В.4 (содержит специальный символ)

3.7. Упорядоченность символов. Символы, образующие программный модуль, являются упорядоченными. Также является упорядоченным любой набор символов, образующий имена, строки и предложения. Это определяется упорядоченностью символов в строке (п. 3.3) и порядком следования строк в модуле.

4. ЭЛЕМЕНТЫ ДАННЫХ

В языке ФОРТРАН имеются семь типов данных: целый, вещественный, двойной точности, комплексный, логический, текстовый и байтовый. Каждый тип данных предназначен для определенных целей и имеет свое внутреннее представление.

4.1. Связь данного с его типом

Каждое имя, используемое для идентификации данного или функции, связано с вполне определенным типом данных. В каждом программном модуле с символическим именем функции, переменной или массива связывается только один тип данных. Эта связь в рамках данного программного модуля устанавливается только однажды и сохраняется для любого другого использования этого символического имени.

Для символического имени тип данных может быть установлен указанием его в объявлении типа (п.п. 8.1.8, 8.1.9) для любого типа данных, кроме текстового. Такое явное объявление аннулирует неявную связь, устанавливаемую для целого и вещественного типов (п. 4.5).

4.2. Свойства данных разных типов. Для данных типа целый, вещественный и двойной точности значение нуль не считается ни положительным, ни отрицательным.

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

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

4.2.3. Тип двойной точности. Данное двойной точности — это более точное приближение вещественного значения. Данное этого типа может принимать положительные, отрицательные и нулевое значения.

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

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

4.2.6. Тип текстовый. Текстовое данное — это последовательность символов. В текстовом данном символ пробела является значащим.

4.2.7. Тип байтовый. Байтовое данное — это целое данное в диапазоне от —128 до +127, а также текстовая константа (Холлерита или буквенно-цифровой литерал (п.п. 4.3.1.7)), представленная в виде одиночного символа.

4.3. Имена данных и процедур. Имена используются как для ссылок на данные и процедуры, так и для любой другой их идентификации. С помощью имен данных идентифицируются константы, переменные, массивы или элементы массивов, а также блоки. С помощью имен процедур идентифицируют функции или подпрограммы.

4.3.1. Константы. Константа является данным, которое всегда определено и не меняется в процессе выполнения программы.

В ФОРТРАНе допускаются следующие виды констант:

— числовые константы: целые, восьмеричные, вещественные, двойной точности, комплексные;

— логические константы;

— текстовые константы.

4.3.1.1. Целая константа. Запись целой константы имеет вид SNN

где S — знак числа;

NN — последовательность от одной до пяти десятичных цифр.

Значение константы этого вида в точности равно числу, изображаемому этой константой в десятичной системе.

Отрицательной константе предшествует знак минус. Знак плюс является необязательным символом. В целую константу не разрешается включать запятые и точки.

Значение целой константы должно находиться в диапазоне от —32768 до 32767.

Примеры:

3564

+14

—26557

4.3.1.2. Восьмеричная константа. Запись восьмеричной константы имеет вид "NN
где NN — последовательность от одной до шести восьмеричных цифр.

Значение константы этого вида в точности равно числу, изображаемому этой константой в восьмеричной системе счисления.

Значение восьмеричной константы должно находиться в диапазоне от 0 до 177777.

Примеры:
"7213
"5

4.3.1.3. Вещественная константа — это либо смешанная дробь, либо смешанная дробь, за которой следует десятичная экспонента, либо целая константа, за которой следует десятичная экспонента.

Основной формой записи вещественной константы является смешанная дробь вида
SP.Q
где S — знак числа;
P — целая часть числа;
Q — дробная часть числа.

Как целая, так и дробная часть есть неотрицательная целая константа. Одна из этих частей (либо целая, либо дробная) может отсутствовать, т. е. представляться пустой последовательностью цифр.

Запись вещественной константы в экспоненциальной форме имеет вид
SP.QE[+]KK или SP.QE—KK или SNNE[+]KK
или SNNE—KK
где S — знак числа;
P — целая часть числа;
Q — дробная часть числа;
K — одна из цифр от 0 до 9;
NN — последовательность десятичных цифр.

Отрицательной вещественной константе предшествует знак минус. Для положительных значений знак плюс, предшествующий константе, необязателен.

Абсолютная величина вещественной константы должна находиться в диапазоне от 0.29**(—38) до 1.7*10**38 (символы «*» и «**» означают знаки операций умножения и возведения в степень соответственно).

Примеры:
15.

.307
—10.685
4.2E+3
2.0E—3
—1.5E6

4.3.1.4. Константа двойной точности. Константа двойной точности записывается как смешанная дробь, за которой следует экспонента двойной точности, либо как целая константа, за которой следует экспонента двойной точности. Экспонента двойной точности записывается и трактуется аналогично десятичной экспоненте, за исключением того, что вместо буквы «E» используется буква «D».

Отрицательной константе двойной точности предшествует знак минус. Для положительных значений знак плюс, предшествующий константе, необязателен.

Абсолютная величина константы двойной точности должна находиться в диапазоне от 0.29*10**(—38) до 1.7*10**38.

Примеры:
1.8D+3
—3.6D—2
7.2D4

4.3.1.5. Комплексная константа. Запись комплексной константы имеет вид (RC, RC) где RC — вещественная константа.

Первое число пары представляет действительную часть комплексной константы, второе — мнимую.

Значение действительной и мнимой части комплексной константы должно находиться в диапазоне, определенном для вещественных констант.

Примеры:
(2.,2.)
(—48.0,+3.7)
(—5.OE+3,.16E+2)

4.3.1.6. Логическая константа. Логические константы «истина» и «ложь» записываются как .TRUE. и .FALSE. соответственно и имеют внутреннее представление «1» и «0». Ненулевое значение всегда рассматривается как «истина».

4.3.1.7. Текстовая константа. Запись текстовой константы имеет три формы:
— Холлерита;
— буквенно-цифровой литерал;
— RADIX—50.

Запись константы Холлерита имеет вид: NHC1C2 . . . CN

где N — целая константа без знака (N>0);
каждое CI — некоторый символ.

Последовательность из N символов, которая следует за буквой H, образует собственно текстовую константу. После буквы H могут быть записаны любые N символов. Символ пробела является значащим. Максимальное число символов равно 255.

Константа холлерита используется в списке фактических параметров оператора вызова подпрограммы (п. 6.2.4), объявлении начальных данных (п. 8.2), объявлении формата (п. 7.2).

Примеры:

4H ABC
16TODAY'S DATE IS:

запись буквенно-цифрового литерала имеет вид:

'C1C2 . . . CN'

где каждое CI — некоторый символ.

Внутри буквенно-цифрового литерала символ апостроф обозначается двумя последовательными апострофами.

Буквенно-цифровой литерал используется в тех же случаях, что и константа Холлерита, а также в операторах останова и паузы (п.п. 6.2.8 и 6.2.9).

Примеры:

'FORMULA'
'WHAT'

запись константы RADIX-50 имеет вид NRC1C2 . . . CN
где N — целая константа без знака (0<N<=12), которая указывает на число символов в константе;
каждое CI — символ из набора символов RADIX—50 (см. табл. 2).

СИМВОЛЫ RADIX-50

Таблица 2

Символ	Название символа
1	2
	пробел
A — Z	прописные буквы латинского алфавита
$	знак денежной единицы
.	точка
0 — 9	цифры от 0 до 9

RADIX—50 — это специальное представление текстовых данных, при котором до трех символов из набора символов RADIX—50 может быть упаковано в одно слово.

Константа RADIX—50 используется только в объявлении начальных данных (п. 8.2).

Примеры:

4RABCD
3R$2Z
5R—T—OA

4.3.2. Переменные. Переменная — это данное, идентифицируемое символическим именем (п. 3.6), значение которого может изменяться в процессе выполнения программы. На это данное можно ссылаться и его можно определять.

4.3.3. Массивы. Массив — это упорядоченный набор данных, имеющий от одного до семи измерений. Массив идентифицируется символическим именем, что позволяет использовать упорядоченный набор данных как единое целое.

Виртуальный массив — это массив, расположенный в реальной памяти за пределами непосредственно адресуемой памяти.

Основные положения, относящиеся к обычному массиву, распространяются и на виртуальный массив.

4.3.3.1. Элемент массива — это одна из компонентов набора данных, образующего массив. Элемент массива идентифицируется указанием имени массива, непосредственно за которым следует дополнительная конструкция, называемая индексом. Индекс указывает на конкретный элемент массива.

На элемент массива можно ссылаться, и его можно определять.

4.3.3.2. Индекс представляет собой заключенный в скобки список индексных выражений, разделенных запятой. Число индексных выражений должно соответствовать объявленному числу измерений массива (п. 8.1.1) за исключением вхождения имени элемента массива в объявление эквивалентности (п. 8.1.6). Идентифицируемый элемент массива определяется при помощи функции линеаризации (п. 8.1.1.1) с использованием вычисленных значений всех индексных выражений.

4.3.3.3. Индексное выражение. В качестве индексного выражения может использоваться любое допустимое арифметическое выражение (п. 5.1), при этом результат вычисления арифметического выражения преобразуется к типу целый.

4.3.4. Процедуры (разд. 9) идентифицируются символическим именем. Язык ФОРТРАН содержит два основных вида процедур: процедуры-функции (или функции) и про-

цедуры-подпрограммы (или внешние подпрограммы).
К функциям относятся:
— основные внешние функции;
— внутренние функции;
— внешние функции.

4.4. Ссылка на функцию производится при помощи указателя функции, состоящего из имени функции, за которым следует список фактических параметров, заключенный в скобки. Если список содержит более одного параметра, то они отделяются друг от друга запятой.

4.5. Правила типов для идентификаторов данных и процедур.

Тип константы определяется ее изображением. С символическим именем, идентифицирующим блок данных или подпрограмму, не связывается никакой тип.

Тип, который связывается с символическим именем, идентифицирующим переменную, массив или внутреннюю функцию, может быть указан при помощи объявления типа (п. 8.1.8). При отсутствии явного объявления типа с символическим именем связывается тип целый, если первая буква этого имени — I, J, K, L, M, N; в противном случае связывается тип вещественный.

Если символическое имя основной внешней функции используется в таком контексте, где оно идентифицирует именно одну из этих функций, то с ним связывается тип соответствующей функции, определенной в приложении 3.

Если в программном модуле содержатся ссылки на внутреннюю функцию, то тип этой функции определяется так же, как и у переменной или массива. Для внешней функции тип определяется либо неявно (по имени функции), либо указывается явно в заголовке функции.

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

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

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

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

Если формальный параметр используется для ссылки на

переменную или на элемент массива, то при задании фактического параметра с этим формальным параметром должно связываться значение того же типа, которое определено по правилам, приведенным в п. 4.5.

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

Если формальный параметр является именем виртуального массива, то соответствующий ему фактический параметр должен являться также именем виртуального массива.

Если фактический параметр ссылается на элемент виртуального массива, то ему должен соответствовать формальный параметр, который является переменной, значение которой не переопределяется в подпрограмме.

5. ВЫРАЖЕНИЯ

ФОРТРАН допускает арифметические и логические выражения, а также отношения. Выражение формируется из операндов и знаков операций.

5.1. Арифметические выражения. Арифметическое выражение представляет собой комбинацию знаков арифметических операций и арифметических операндов. Как выражение, так и входящие в него операнды идентифицируют значение типа целый, вещественный, двойной точности или комплексный.

Знаки арифметических операций приведены в табл. 3.

Таблица 3

Знак операции	Представляемая операция
1	2
+	сложение
—	вычитание
*	умножение
/	деление
**	возведение в степень

Арифметические операции сложения и вычитания могут быть одноместными и двуместными. В случае одноместных

операций сложения и вычитания подразумеваемым первым арифметическим операндом является нуль.

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

Первичное арифметическое выражение — это либо арифметическое выражение, взятое в скобки, либо константа, либо ссылка на переменную, либо ссылка на элемент массива, либо ссылка на функцию.

Множитель — это либо первичное арифметическое выражение, либо конструкция вида:
первичное арифметическое выражение**первичное арифметическое выражение
Терм — это либо множитель, либо конструкция одного из видов
терм/множитель
или
терм*множитель

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

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

Арифметическое выражение — это либо простое арифметическое выражение, либо терм со знаком, либо одна из двух конструкций, за которой непосредственно следует знак плюс или минус, за которым, в свою очередь, следует простое арифметическое выражение.

Первичное арифметическое выражение любого типа может возводиться в степень, показателем которой является первичное арифметическое выражение. Множитель, получающийся в результате, имеет тот же тип, что и возводимое в степень первичное арифметическое выражение за исключением случая, когда первичное арифметическое выражение типа вещественный возводится в степень, показателем которой является первичное арифметическое выражение типа двойной точности: в этом случае получающийся множитель имеет тип двойной точности. Первичное арифметическое выражение, имеющее отрицательное значение, можно возводить только в целую степень. Первичное арифметическое выражение, имеющее нулевое значение, нельзя возводить в нулевую степень. Допустимые сочетания типов данных основания и показателя степени для операции возведения в степень приведены в табл. 4.

Таблица 4

Тип основания	Тип показателя степени			
	целый	вещественный	двойной точности	комплексный
целый	да	нет	нет	нет
вещественный	да	да	да	нет
двойной точности	да	да	да	нет
комплексный	да	нет	нет	нет

Если в арифметическом выражении все операнды имеют один и тот же тип, то выражение будет иметь тот же самый тип. Если в выражении имеются операнды различных типов, то выражение будет иметь тип операнда более высокого приоритета. Типы данных и их приоритеты приведены в табл. 5.

Таблица 5

Тип	Приоритет
1	2
комплексный	первый
двойной точности	второй
вещественный	третий
целый	четвертый

Арифметическое выражение будет иметь тип:
- целый, если все операнды, входящие в выражение, типа целый или типа целый и логический. Восьмеричные константы и логические данные рассматриваются как данные типа целый;
- вещественный, если все операнды, входящие в выражение, типа вещественный или типа вещественный и целый. Операнды типа целый преобразуются к типу вещественный;
- двойной точности, если все операнды, входящие в выражение, типа двойной точности или типа двойной точности, вещественный и целый. Операнды типа вещественный и целый преобразуются к типу двойной точности;
- комплексный, если все операнды, входящие в выражение, типа комплексный или типа комплексный, двойной точно-

сти, вещественный и целый. Операнды типа двойной точности и целый преобразуются к типу вещественный и используются как действительная часть комплексного числа. Мнимая часть числа в этом случае равна нулю.

5.2. Отношения. Отношение состоит из двух арифметических выражений, разделенных знаком операции отношения, и принимает значение «истина» или «ложь» в зависимости от выполнения или невыполнения этого отношения. Знаки операций отношения приведены в табл. 6.

Допустимыми знаками операций отношения, содержащего одно или оба арифметических выражения типа комплексный, являются .EQ и .NE., результат при этом будет иметь тип комплексный. Если в отношении арифметические выражения имеют разный тип, то результат будет иметь тип выражения более высокого приоритета (см. табл. 6).

Таблица 6

Знак операции	Представляемая операция
1	2
.LT.	Меньше
.LE.	Меньше или равно
.EQ.	Равно
.NE.	Не равно
.GT.	Больше
.GE.	Больше или равно

5.3. Логические выражения. Логическое выражение формируется из знаков логических операций и логических операндов и принимает значение «истина» или «ложь».

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

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

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

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

Логическое выражение — это либо логический терм, либо конструкция вида: логическое выражение .OR. логическое выражение.

Знаки логических операций приведены в табл. 7.

Таблица 7

Знак операции	Представляемая операция
1	2
.AND.	Логическое умножение
.OR.	Логическое сложение
.NOT.	Логическое отрицание
.XOR.	Исключающее ИЛИ
.EQV.	Логическая эквивалентность

Значение выражения A.AND.B истинно, если значения операндов A и B истинны.

Значение выражения A.OR.B истинно, если значение одного операнда A или B или значения обоих операндов истинны.

Значение выражения .NOT.B истинно, если значение операнда B ложно.

Значение выражения A. XOR.B истинно, если операнды A и B имеют разные значения.

Значение выражения A.EQV.B истинно, если операнды A и B имеют одинаковые значения.

Если в логическом выражении один из операндов или оба имеют тип целый, то операция будет осуществляться над соответствующими разрядами внутреннего (двоичного) представления операндов; при этом результат будет иметь значение «истина», если его значение не равно нулю, в противном случае — значение «ложь».

5.4. Вычисление выражений

Вычисление выражений должно производиться с учетом следующих ограничений.

Если два операнда соединены знаком операции, то порядок вычисления этих операндов произвольный.

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

Не может быть вычислен никакой операнд, значение которого математически не определено. Перечень всех операций и порядок, в котором они выполняются, приведен в табл. 8.

Таблица 8

Знак операции	Приоритет выполнения операции
1	2
**	первый
*, /	второй
+, —	третий
Операции отношения	четвертый
.NOT.	пятый
.AND.	шестой
.OR.	седьмой
.XOR., .EQV.	восьмой

Операции, имеющие одинаковый приоритет, выполняются последовательно слева направо.

Для изменения обычного порядка выполнения операций могут использоваться скобки. Выражение, взятое в скобки, вычисляется первым. Вычисление выражения, взятого в скобки, подчиняется обычному порядку.

6. ОПЕРАТОРЫ

Операторы — это выполняемые предложения, т. е. операторы определяют какие-либо действия.

В языке ФОРТРАН различаются три типа операторов:
— операторы присваивания;
— операторы управления;
— операторы ввода—вывода.

6.1. Операторы присваивания определяют значение переменной или элемента массива.

Существуют три типа операторов присваивания:
— арифметический оператор присваивания;
— логический оператор присваивания;
— оператор предписания.

6.1.1. Арифметический оператор присваивания имеет вид
V=E
где V — имя переменной или имя элемента массива любого типа, отличного от логического;
E — арифметическое выражение.

Знак равенства здесь не означает «равно», а определяет замену.

Арифметический оператор присваивания присваивает значение арифметического выражения, расположенного справа от знака равенства, переменной или элементу массива, находящемуся слева от знака равенства. Если переменная имела до этого какое-либо значение, то оно теряется.

Результатом выражения, стоящего справа от знака равенства должно быть значение, удовлетворяющее требованиям типа переменной или элемента массива, которым оно присваивается.

Если тип переменной или элемента массива, находящегося слева от знака равенства, точно такой же, как у значения выражения справа, оператор непосредственно выполняет присваивание. Если типы различаются, то перед присваиванием значение выражения преобразуется к типу объекта, находящегося слева от знака равенства.

Примеры:
BETA=—1./(2.*A)+3.**2
P=3.14159
SUM=SUM+1

6.1.2. Логический оператор присваивания имеет вид V=E
где V — имя логической переменной или имя элемента логического массива;
E — логическое выражение.

Логический оператор присваивания подобен арифметическому оператору присваивания, но он оперирует с логическими данными.

Выполнение такого оператора заключается в вычислении логического выражения E и присваивании вычисленного значения логическому объекту V.

Примеры:
ABS=.TRUE.
COM=X.GT.85.AND..NOT.Y

6.1.3. Оператор предписания имеет вид ASSIGN K TO I
где K — метка оператора из того же программного модуля, что и оператор предписания;
I — имя переменной типа целый.

Оператор предписания используется для присваивания целой переменной метки оператора. Затем эта переменная может быть использована в последующем операторе перехода по предписанию (п. 6.2.1.2).

Оператор предписания присваивает переменной метку оператора по правилам арифметического оператора присваивания.

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

Примеры:

ASSIGN 100 TO NUMBER

оператор ASSIGN ставит в соответствие переменной NUMBER метку оператора 100. После этого операции над этой переменной такие, как NUMBER=NUMBER+1 недопустимы.

NUMBER=10

оператор присваивает переменной NUMBER значение 10. Теперь эта переменная не может быть использована в операторе перехода по предписанию.

ASSIGN 100 TO ERAS

ERAS должна быть определена как целая переменная.

6.2. Операторы управления. Операторы, как правило, выполняются в том порядке, в каком они записаны. Однако иногда бывает необходимо прервать или изменить последовательный порядок выполнения программы. Это осуществляется с помощью операторов управления. В языке ФОРТРАН имеются восемь типов операторов управления:

— операторы перехода;
— условный арифметический оператор;
— условный логический оператор;
— оператор вызова подпрограммы;
— оператор возврата;
— оператор цикла;
— оператор продолжения;
— операторы останова и паузы.

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

6.2.1. Операторы перехода передают управление внутри программного модуля либо каждый раз на один и тот же оператор, либо на один из операторов, принадлежащих некоторому множеству, в зависимости от значения указанного выражения.

Имеются три типа операторов перехода:

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

6.2.1.1. Безусловный оператор перехода имеет вид:

GOTO K

где K — метка оператора.

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

Примеры:

GO TO 5

GO TO 99999

6.2.1.2. Оператор перехода по предписанию имеет вид:

GO TO I [[,] (K1[, K2] . . .)]

где I — имя переменной типа целый;
каждое KJ — метка оператора.

К моменту выполнения оператора перехода по предписанию переменной I должно быть присвоено текущее значение предшествующим выполнением оператора предписания (п. 6.1.3); этим значением должна быть одна из меток списка, заключенного в скобки. Результат выполнения такого оператора перехода состоит в том, что следующим будет выполняться оператор, помеченный этой меткой.

При отсутствии списка меток действие оператора перехода по предписанию аналогично выполнению безусловного оператора перехода.

Оператор перехода по предписанию и соответствующий оператор предписания должны быть расположены внутри одного и того же программного модуля. Операторы, которым передается управление, также должны быть расположены в том же программном модуле.

Примеры:

GO TO INDEX (10, 30, 7, 2)

GO TO MAX

6.2.1.3. Вычисляемый оператор перехода имеет вид

GO TO (K1[, K2] . . .) [,] I

где каждое KJ — метка оператора;
I — арифметическое выражение типа целый, вещественный или двойной точности (1<=I<=N, N — число меток).

Вычисляемый оператор перехода вычисляет значение I, преобразует его значение к целому типу и передает управление на метку оператора, порядковый номер которой в списке меток операторов равен полученному значению выражения. Если значение выражения меньше единицы или больше N, управление будет передано оператору, следующему за оператором GO TO.

Примеры:

GO TO (2, 5, 7) IND

GO TO (10, 20, 30, 40, 50), KOM

6.2.2. Условный арифметический оператор имеет вид
IF (E) K1, K2, K3

где E — арифметическое выражение типа целый, вещественный или двойной точности;
каждое KJ — метка оператора.

В записи оператора должны присутствовать все три метки. Эти метки необязательно должны относиться к трем разным операторам.

При выполнении условного арифметического оператора прежде всего вычисляется выражение E, а затем управление передается оператору с меткой K1, K2 или K3, если значение выражения E соответственно меньше, равно или больше нуля.

Пример.
IF (SUMI—SUM) 5, 5, 10

Этот оператор передает управление оператору с меткой 5, если значение переменной SUMI меньше или равно значению переменной SUM. Управление передается оператору с меткой 10, если значение переменной SUMI больше значения переменной SUM.

6.2.3. Условный логический оператор имеет вид IF (E) S

где E — логическое выражение;
S — любой оператор, кроме оператора цикла и условного логического оператора.

При выполнении этого оператора сначала вычисляется логическое выражение E. Если E принимает значение «истина», то выполняется оператор S. Если E принимает значение «ложь», то оператор выполняется так, как если бы он был оператором продолжения (п. 6.2.7), т. е. в этом случае оператор S фактически не выполняется.

Пример.
IF (J.GT.4.OR.J.LT.I) GO TO 200

6.2.4. Оператор вызова подпрограммы имеет вид
CALL S [([A1][[,A2]] . . .)]

где S — имя подпрограммы на языке ФОРТРАН или АССЕМБЛЕР;
каждое AI — **фактический параметр.**

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

Параметры, указанные в операторе, должны по своему количеству, порядку следования и типу данных соответствовать формальным параметрам в определении подпрограммы. Параметры оператора CALL могут быть арифметическими выражениями, буквенно-цифровыми литералами или именами подпрограмм.

Примеры:
CALL PROCENT (X, 2.75, P(3), 3.14+Y)
CALL SUM (3, 5)

6.2.5. Оператор возврата имеет вид RETURN

Оператор возврата используется только в модуле-процедуре (модуле-подпрограмме и модуле-функции) и предназначен для того, чтобы отмечать его логический конец. Результат его выполнения состоит в возврате управления в тот модуль, из которого было произведено обращение к рассматриваемому модулю-процедуре (оператору, следующему непосредственно за оператором вызова).

Пример.
SUBROUTINE SUM (A, B, C)
C=A+B
RETURN
END

6.2.6. Оператор цикла имеет вид DO N [,] I=M1, M2 [, M3]

где N — метка оператора;
I — имя переменной типа целый;
каждое MJ — арифметическое выражение типа целый.

Оператор, помеченный меткой N и называемый закрывающим оператором тела цикла, должен находиться в том же программном модуле, что и рассматриваемый оператор цикла, и физически должен помещаться после него. Закрывающий оператор не может быть оператором перехода, возврата, останова, цикла, условным арифметическим оператором, а также условным логическим оператором, содержащим какой-либо из указанных операторов.

Переменная I называется управляющей переменной; M1 называется начальным параметром, M2 — конечным параметром и M3 — параметром приращения. Параметр приращения может быть опущен, если он равен единице. Значения начального параметра, конечного параметра и приращения не могут изменяться внутри тела цикла, но допускается обращение к управляющей переменной как к обычной переменной.

Под телом оператора цикла понимается последовательность операторов, начиная с первого по порядку оператора, следующего за рассматриваемым оператором цикла, и кон-

чая его закрывающим оператором. В частности, если тело одного оператора цикла содержит другой оператор цикла, то тело этого другого оператора цикла должно быть подмножеством тела первого.

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

Оператор цикла служит для задания цикла в программе.

Действия, порождаемые выполнением оператора цикла, следующие:

1) вычисляются выражения M1, M2, M3 для определения значений начального, конечного параметров и параметра приращения;

2) управляющей переменной присваивается значение начального параметра;

3) выполняется тело оператора цикла;

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

5) если значение управляющей переменной, полученное в результате выполнения шага 4), находится в диапазоне между начальным и конечным параметром, то повторяются описанные выше действия, начиная с указанных в 3) с учетом того, что под телом цикла понимается тело того оператора цикла, управляющая переменная которого позже всех получила приращение. Если значение управляющей переменной оказалось вне указанного диапазона, то оператор цикла считается завершенным и значение его управляющей переменной становится неопределенным;

6) если имеются другие операторы цикла, тела которых заканчиваются указанным закрывающим оператором, то значение управляющей переменной того из этих операторов цикла, выполнение которого началось позже всех, изменяется на значение, представленное соответствующим параметром приращения, и повторяются действия, указанные в 5), до тех

пор, пока не будут завершены все операторы цикла, тела которых заканчиваются указанным закрывающим оператором. После этого выполняется оператор, следующий за закрывающим оператором;

7) после выхода из тела оператора цикла в результате выполнения оператора перехода, условного арифметического оператора или условного логического оператора, содержащего указанные операторы, т. е. способом, отличным от завершения оператора цикла, значение управляющей переменной этого оператора цикла определено и равно последнему ее значению, достигнутому по правилам предыдущих шагов.

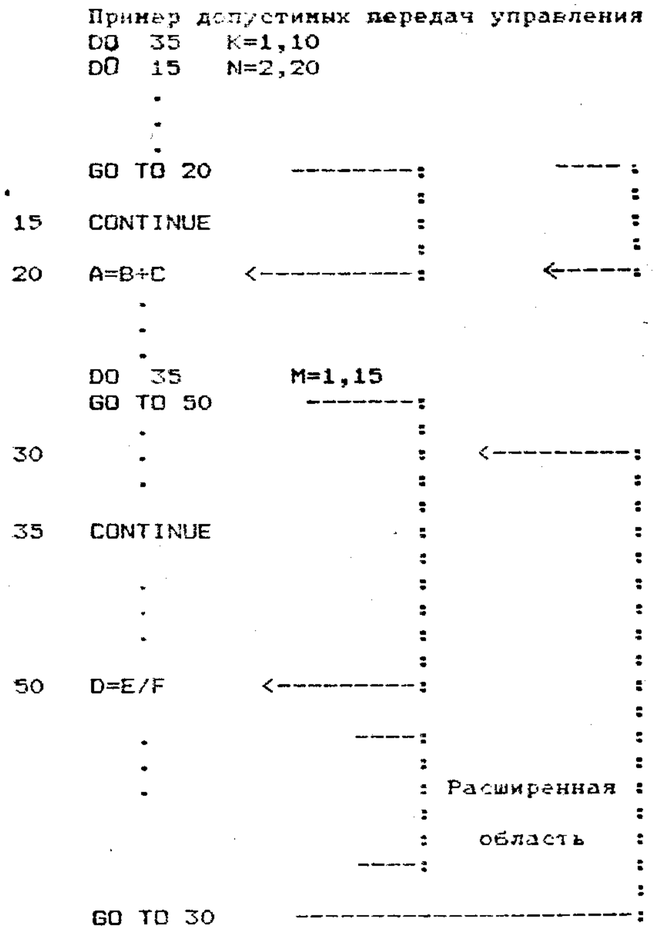
Число выполнений тела оператора цикла, называемое числом итераций, определяется как наибольшее абсолютное целое значение выражения (M2—M1)/M3+1
если число итераций равно нулю или отрицательно, цикл выполняется только один раз.

Оператор цикла имеет расширенное тело, если выполнены следующие условия:

1) в правильном гнезде внутри тела самого внутреннего оператора цикла имеется оператор перехода, условный арифметический оператор, или условный логический оператор, содержащий указанные операторы, который может передать управление в область расположенную вне этого гнезда;

2) вне гнезда имеется оператор перехода, условный арифметический оператор или условный логический оператор (содержащий указанные операторы), который с учетом всех возможных последовательностей выполнения операторов в данном программном модуле может быть выполнен после оператора, указанного в 1), что может привести к возврату управления в тело самого внутреннего оператора цикла того же правильного гнезда.

Если оба эти условия выполнены, то расширенное тело определяется как тело оператора цикла вместе с его расширением, т. е. множеством всех операторов, которые могут быть выполнены между всеми параметрами операторов, передающих управление, первый из которых удовлетворяет условию, указанному в 1), а второй — условию, указанному в 2). Первый оператор пары не включается в расширение тела, а второй включается. Оператор перехода, условный арифметический оператор или условный логический оператор, содержащий указанные операторы, не могут приводить к передаче управления в само тело оператора цикла — за исключением того случая, когда эти операторы выполняются как часть расширенного тела данного оператора цикла. Кро-



ме того, область расширения тела оператора цикла не может содержать оператор цикла (того же самого программного модуля), который в свою очередь имеет расширенное тело. Если в теле оператора цикла встречается обращение к процедуре, то действия, определяемые этой процедурой, считаются временно (т. е. на время выполнения этой процедуры) включенными в тело оператора цикла.

6.2.7. Оператор продолжения имеет вид CONTINUE

Этот оператор не влияет на последовательность выполнения операторов программы. Он используется, главным образом, как закрывающий оператор в теле оператора цикла.

Пример.

```
DO 5 I=1,5
J=I**2
5 CONTINUE
```


6.2.8. Оператор паузы имеет вид PAUSE [N]

где N — последовательность десятичных или восьмеричных цифр или буквенно-цифровой литерал. Последовательности восьмеричных цифр должны предшествовать кавычки.

Действие оператора PAUSE зависит от режима выполнения программы. Если она выполняется в пакетном режиме, то сообщение PAUSE — — N записывается в регистрационный файл программы BATCH (см. [1]), и выполнение программы не прерывается. Если программа выполняется в интерактивном режиме, то сообщение PAUSE — — N выводится на терминал, и выполнение программы прерывается. После подачи <ВК> выполнение программы возобновляется с первого оператора, следующего за PAUSE.

Примеры:

```
PAUSE "12345
PAUSE 'PROVERKA'
```

6.2.9. Оператор останова имеет вид STOP[N]

где N — последовательность десятичных или восьмеричных цифр или буквенно-цифровой литерал. Последовательности восьмеричных цифр должны предшествовать кавычки.

Оператор STOP выводит на терминал сообщение STOP — — N (если оно определено) и выполнение программы прекращается.

Примеры:

```
STOP 255
STOP 'END OF PROGRAM'
```

7. СРЕДСТВА ОБМЕНА ДАННЫМИ

7.1. Общая характеристика и назначение операторов ввода-вывода

Операторы ввода-вывода используются для организации обмена информацией между основной памятью и устройствами ввода-вывода.

Операторы ввода-вывода делятся на две группы. К первой группе относятся основные операторы ввода-вывода, ко второй — вспомогательные.

Основные операторы — это операторы, в результате выполнения которых передаются записи из файла во внутреннюю память и обратно. Они делятся на операторы последовательного доступа и операторы прямого доступа.

Операторы ввода-вывода последовательного доступа используются для последовательного запоминания и последовательной выборки данных. Эти операторы обрабатывают фай-

лы (упорядоченный набор записей) как на устройствах последовательного доступа, так и на устройствах прямого доступа.

Операторы ввода-вывода прямого доступа используются для запоминания и выборки данных в порядке, указанном пользователем. Эти операторы обрабатывают файлы только на устройствах прямого доступа.

Записи могут быть форматными и бесформатными. Форматная запись состоит из последовательности символов КОИ-7. Передача такой записи требует ссылки на спецификацию формата (п. 7.2), которая определяет необходимые преобразования и размещение записи. Число записей, передаваемых при выполнении операторов форматного ввода или вывода, зависит от списка ввода-вывода и указанной спецификации формата. Бесформатная запись состоит из последовательности двоичных данных. Когда выполняется оператор форматного или бесформатного ввода, требуемые записи на указанном устройстве должны быть соответственно форматными или бесформатными.

К основным операторам ввода-вывода относятся также операторы ввода-вывода с преобразованием по списку. Операторы ввода-вывода с преобразованием по списку — это операторы форматного ввода-вывода последовательного доступа без использования ссылки на спецификацию формата; преобразование данных происходит в соответствии с типом элементов списка ввода-вывода. Наличие списка ввода-вывода в операторах с преобразованием по списку является обязательным. Признаком преобразования по списку является символ «*».

Ссылка на спецификацию формата задается с помощью указателя формата — метки объявления формата (п. 7.2) или имени массива, содержащего спецификацию формата.

Основные операторы ввода-вывода могут содержать параметры END=S и/или ERR=S, где S — метка оператора. Оператор, помеченный меткой S, должен находиться в том же программном модуле, что и оператор ввода-вывода.

Если при выполнении оператора ввода, содержащего параметр END=S, встретится запись «Конец файла», то следующим будет выполняться оператор, помеченный меткой S. Если такой параметр отсутствует, будет зафиксировано состояние ошибки.

Если при выполнении оператора ввода-вывода, содержащего параметр ERR=S, возникает состояние ошибки при передаче записи с устройства в память или обратно, управление передается оператору, помеченному меткой S. Если такой параметр отсутствует, выполнение программы будет прекращено.

Вспомогательные операторы — это операторы, осуществляющие различные подготовительные операции для проведения ввода-вывода. К этой группе операторов относятся: операторы перемотки, сдвига назад, поиска записи, а также разметки, описания, открытия и закрытия файла.

7.1.1. Устройства ввода-вывода

Операторы ввода-вывода обращаются к устройствам ввода-вывода с помощью номеров логических устройств (табл. 9). Номер логического устройства задается в виде константы или переменной типа целый.

Таблица 9

Номер логического устройства	Устройство ввода-вывода	Обозначение устройства
1	2	3
1	системный диск	DK:
2	стандартное устройство	—
3	стандартное устройство	—
4	стандартное устройство	—
5	терминал (ввод)	TT:
6	построчно-печатающее устройство	LP:
7	терминал (вывод)	TT:
8	перфоленточное устройство (ввод)	PC:
9	перфоленточное устройство (вывод)	PC:

В некоторых операторах ввода-вывода не указывается номер логического устройства, эти операторы используют номер логического устройства, который неявно определяется операционной системой.

7.1.2. Списки ввода-вывода. Список ввода определяет имена переменных, массивов и элементов массивов, которым присваиваются значения при вводе. Список вывода определяет константы, переменные, массивы, элементы массивов и выражения, значения которых передаются при выводе.

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

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

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

Спецификация цикла имеет вид

I=M1, M2[,M3]

где I, M1, M2, M3 — определяются так же, как и в операторе цикла (п. 6.2.6).

Область действия спецификации цикла — это список, входящий в состав списка с циклом; для списков ввода I, M1, M2, M3 могут встречаться внутри этой области только в индексах.

Элементы списка считаются упорядоченными в соответствии с их вхождением в список при его просмотре слева направо.

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

Объявление формата имеет вид

FORMAT (Q1T1Z1T2Z2...ZN—1TNQ2)

где (Q1T1Z1T2Z2...ZN—1TNQ2) — спецификация формата;
каждое ...QI — серия дробных черт или пусто;
каждое TJ — описатель поля или группа описателей полей;
каждое ZK — разделитель полей;
N — может быть равно нулю.
Каждое объявление формата должно быть помечено.

7.2.1. Описатели полей формата имеют вид

SRFW.Y

SREW.Y

SRGW.Y

SRDW.Y

RIW

ROW

RLW

RAW

NHC1C2...CN

'...'

NX

TN

Q

:

$

где F, E, G, D, I, O, L, A, H, X, T — символы, указывающие способ преобразования и редактирования при переходе от внутреннего представления записей к внешнему и обратно и являющиеся кодами преобразований;

W и N — отличные от нуля целые константы без знака, указывающие ширину поля для внешней последовательности символов;

Y — целая константа без знака, указывающая количество цифр в дробной части числа, изображаемого внешней последовательностью символов (за исключением кода преобразования G);

R — либо пусто, либо отличная от нуля целая константа без знака, называемая счетчиком повторений, который указывает, сколько раз должен быть повторен следующий за ним описатель поля;

S — либо пусто, либо конструкция, обозначающая предписатель масштабного множителя (п. 7.2.4);
Каждое CI — один из символов КОИ-7;
R, W, Y — не должны превышать 255.

Для описателей вида W.Y обязательно должно быть указано Y даже в том случае, когда оно равно нулю, W должно быть больше либо равно Y.

Описатели числовых полей I, O, F, E, G, D используются для задания ввода-вывода данных типа целый, вещественный, двойной точности и комплексный.

Описатель поля L используется для задания ввода-вывода данных типа логический.

Описатели полей A, H, '...' используются для задания текстовых данных.

Описатели полей X, T, Q, :, $ используются при редактировании данных.

Описатели полей, записанные в виде кодов преобразований F, E, G, D, I, O, L, A используют значения W и Y по умолчанию согласно табл. 10.

Таблица 10

Описатель поля	Тип элемента списка	W	Y
1	2	3	4
I или O	INTEGER*2	7	
I или O	INTEGER*4	12	
L	LOGICAL	2	
F, E, G или D	REAL, COMPLEX	15	7
F, E, G или D	DOUBLE PRECISION	25	16
A	LOGICAL*1	1	
A	INTEGER*2	2	
A	LOGICAL*4, INTEGER*4	4	
A	REAL, COMPLEX	4	
A	DOUBLE PRECISION	8	

Повторение группы описателей полей или разделителей полей достигается заключением их в скобки и помещением перед левой скобкой счетчика повторений группы. Такая форма называется основной группой.

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

Внутреннее представление внешних полей соответствует внутреннему представлению констант соответствующих типов (п.п. 4.2 и 4.3.1).

Спецификация формата может содержать большее количество описателей полей, чем содержится элементов списка в соответствующем операторе ввода-вывода; избыточное число описателей игнорируется.

Для всех описателей числовых полей при вводе ведущие пробелы являются незначащими, а остальные пробелы трактуются как нули. Знак плюс может быть опущен. Поле, состоящее из одних пробелов, трактуется как нуль. Для преобразования F, E, G, D точка, присутствующая в поле ввода, отменяет задание точки в описателе поля.

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

Если число символов, порождаемых преобразованием, больше ширины поля, то поле вывода заполняется символами «*».

7.2.1.1. Описатель I. Описатель числового поля IW указывает, что внешнее поле — это целая константа, занимающая W позиций. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа целый или логический.

Во внешнем поле ввода последовательность символов, с учетом трактовки пробелов, должна представлять целую константу.

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

Примеры:

Ввод

Описатель поля	Внешнее поле	Внутреннее значение
I5	␣␣123	123
I3	—44	—44

Вывод

Описатель поля	Внутреннее значение	Внешнее поле
I4	**13**	␣␣13
I2	156	**

7.2.1.2. Описатель O. Описатель числового поля OW указывает, что внешнее поле — это восьмеричная константа, занимающая W позиций. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа целый или логический.

Во внешнем поле ввода последовательность символов, с учетом трактовки пробелов, должна представлять восьмеричную константу.

Внешнее поле вывода состоит, если это необходимо, из пробелов, за которыми следует восьмеричная константа, представляющая величину внутреннего данного.

Примеры:

Ввод

Описатель поля	Внешнее поле	Внутреннее значение
03	**145**	145
04	**__27**	27

Вывод

Описатель поля	**Внутреннее значение**	**Внешнее поле**
02	**12**	**14**
05	129	__201

7.2.1.3. Описатель F. Описатель числового поля FW.Y указывает, что внешнее поле занимает W позиций, а дробная часть состоит из Y цифр. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа вещественный.

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

целая константа;

E, за которой следует целая константа;

D, за которой следует целая константа.

Экспонента, содержащая D, эквивалентна экспоненте, содержащей E.

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

Ширина поля W должна учитывать наличие знака минус, если он имеется, как минимум — одной цифры слева от точки, самой точки и наличие D цифр справа от точки; таким образом, величина W должна быть больше или равна величине (Y+3).

Примеры:

Ввод

Описатель поля	Внешнее поле	Внутреннее значение
F7.3	1234567	1234.567
F8.3	14.4E+_**2**	1440.0

Вывод

Описатель поля	Внутреннее значение	Внешнее поле
F5.2	1.2345	_1.23
F5.2	1234.567	*****

7.2.1.4. Описатель E. Описатель числового поля EW.Y указывает, что внешнее поле занимает W позиций, а дробная часть состоит из Y цифр. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа вещественный. Форма внешнего поля ввода такая же, что и для преобразования F.

Стандартная форма внешнего поля вывода для масштабного множителя, равного нулю, состоит из знака минус, если величина отрицательная (знак плюс не указывается), нуля (он тоже необязателен), точки, Y цифр справа от точки и порядка числа в виде E+NN или E—NN
где NN — двузначная целая константа.

Величина W должна быть больше или равна величине (Y+7).

Примеры:

Ввод

Описатель поля	Внешнее поле	Внутреннее значение
E8.3	1.2E__2	120.0
E8.3	2.34D+_1	2.34E+_1
E6.2	534.23	534.23

Вывод

Описатель поля	Внутреннее значение	Внешнее поле
E9.3	123.45678	0.123E +03
E9.2	—0.0014	—0.14E+ —02
E5.2	21.7	*****

7.2.1.5. Описатель D. Описатель числового поля DW.Y указывает, что внешнее поле занимает W позиций, а дробная часть состоит из Y цифр. Значение элемента списка является или (после ввода) должно являться во внутреннем представлении данным типа двойной точности.

Основная форма внешнего поля ввода та же самая, что и для преобразования F, за исключением того, что вводимые данные преобразуются и присваиваются объекту с двойной точностью.

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

Примеры:

Ввод

Описатель поля	Внешнее поле	Внутреннее значение
D7.2	_123.45	123.45D__0
D7.1	1.4D__1	1.4D__1

Вывод

Описатель поля	Внутреннее значение	Внешнее поле
D9.3	2.35	0.235D+_1
D5.3	**77.1**	*****

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

Основная форма внешнего поля ввода та же самая, что и для преобразования F.

Метод представления внешней последовательности символов при выводе зависит от величины данного, подвергаемого преобразованию.

Пусть M — величина внутреннего данного. Соответствие между M и эквивалентным применяемым методом преобразования задается согласно табл. 11.

Таблица 11

Величина данного	Преобразование
1	2
M<0.1	EW.Y
0.1<=M<1	F(W—4).Y ; 4X
1<=M<10	F(W—4).(Y—1) ; 4X
.	.
.	.
10**(I—2)<=M<10**(I—1)	F(W—4).1 ; 4X
10**(I—1)<=M<10**(I)	F(W—4).0 ; 4X
10**(I)<=M	EW.Y

Действие описателя NX см. в п. 7.2.1.11.

Примеры:

Вывод

Описатель поля	Внутреннее значение	Внешнее поле
G8.2	0.024	0.24E—01
G10.3	8.124	—8.12___

7.2.1.7. Описатель L. Описатель логического поля LW указывает, что внешнее поле занимает W позиций как последовательность символов, вид которой определен далее. Элемент списка является или (после ввода) должен являться во внутреннем представлении данным типа логический или целый.

Внешнее поле ввода должно состоять из, возможно, пробелов, за которыми следует символ T либо символ F, за которым, возможно, следуют остальные символы, изображающие значение «истина» либо «ложь» соответственно.

Внешнее поле вывода состоит из (W — 1) пробелов, за которыми следует символ T или символ F, если значение внутреннего данного — «истина» или «ложь» соответственно.

Примеры:

Ввод

Описатель поля	Внешнее поле	Внутреннее значение
L4	TRUE	.TRUE.
L2	—F	.FALSE.

Вывод

Описатель поля	Внутреннее значение	Внешнее поле
L3	.TRUE.	—T
L7	.FALSE.	____F

7.2.1.8. Описатель A. Описатель текстового поля AW указывает, что при вводе прочитывается W символов и передаются в элемент, определяемый списком ввода; а при выводе из элемента списка вывода выбираются W символов.

Максимальное число символов, которое может запомниться в переменной или в элементе массива, зависит от типа данных этого элемента (табл. 12).

Таблица 12

Элемент списка ввода-вывода	Максимальное число символов
1	2
LOGICAL*1	1
LOGICAL*4	4
INTEGER*2	2
INTEGER*4	4
REAL	4
DOUBLE PRECISION	8
COMPLEX	8

Если ширина поля W больше либо равна максимальному числу символов M, то при вводе из внешнего поля будут взяты самые правые M символов, избыточные левые символы при этом теряются. Если ширина поля W меньше максимального числа символов M, то при вводе во внутреннем представлении будет W символов, сдвинутых влево, за которыми будет следовать (M — W) пробелов.

Если ширина поля W больше максимального числа символов M, то при выводе внешнее поле будет состоять из (W—M) пробелов, за которыми следуют M символов из внутреннего представления. Если ширина поля W меньше либо равна максимальному числу символов M, то при выводе внешнее поле будет состоять из W самых левых символов из внутреннего представления.

Примеры:

Ввод

Описатель поля	Внешнее поле	Внутреннее значение
A5	ABCDE	DE (INTEGER*2)
A3	ABC	ABC— (REAL)

Вывод

Описатель поля	Внутреннее значение	Внешнее поле
A5	AB	——AB
A3	ABCD	ABC

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

За описателем поля NH необязательно должен следовать разделитель полей—запятая.

Пример.

```
      ACCEPT    10
10    FORMAT    (6HINPUT)
      TYPE      20
20    FORMAT    (7HOUTPUT)
        .
        .
        .
```

Оператор ACCEPT располагает вводимые данные с клавиатуры терминала. В описателе поля 6H символы «INPUT» могут быть заменены любыми другими шестью символами.

Оператор TYPE передает символы «OUTPUT» из описателя поля 7H на терминал.

7.2.1.10. Описатель '...'. Описатель текстового поля '...' указывает, что в качестве текстовой информации, подлежащей передаче при вводе или выводе, берутся символы, заключенные в апострофы.

За описателем поля '...' необязательно должен следовать разделитель полей—запятая.

Пример.

```
      TYPE 1
1     FORMAT ('INPUT — OUTPUT')
```

оператор TYPE передает символы «INPUT—OUTPUT» из описателя поля '...' на терминал.

7.2.1.11. Описатель X. Описатель поля пробелов NX вызывает при вводе пропуск N символов из внешней вводимой записи, а при выводе во внешнюю выводимую запись-вставку N пробелов.

За описателем поля NX необязательно должен следовать разделитель полей—запятая.

Пример.

```
      TYPE 2
2     FORMAT ('PRINT', 3X, 'WRITE')
```

оператор TYPE передает символы на терминал в следующем виде

PRINT___WRITE

7.2.1.12. Описатель T. Описатель поля табуляции TN (где N—отличная от нуля целая константа без знака, ограниченная числом символов во внешнем поле) указывает на номер позиции N во внешнем поле ввода-вывода, начиная с которой осуществляется передача символов.

Если устройством вывода является терминал или построчно-печатающее устройство, описатель поля TN указывает на (N—1)-ю позицию внешнего поля вывода, в соответствии с которой осуществляется запись первого из передаваемых символов.

Примеры:

Внешнее поле вывода имеет вид

ABC___XYZ

Оператор ввода, связанный с объявлением формата

1 FORMAT (T7,A3,T1,A3)

вводит символы в следующем порядке: сначала XYZ, потом — ABC.

Если программный модуль содержит предложения:

```
      TYPE 2
```

```
2     FORMAT (T10, 'PAGE')
```
то оператор TYPE вызовет печать на терминал строки
```
------PAGE
      /!\
```
9 позиция

7.2.1.13. Описатель Q используется для подсчета символов, которые не были переданы во время операции ввода.

Элементу списка ввода, соответствующему описателю Q, присваивается значение, равное числу непереданных символов внешнего поля ввода.

Элемент списка вывода, соответствующий описателю Q, игнорируется.

Элемент списка ввода-вывода, соответствующий описателю Q, должен иметь тип целый или логический.

Пример.

```
      READ (1,2) S, I
2     FORMAT (F3.1,Q)
```
Внешнее поле ввода содержит: 12345.

Элементу списка ввода I, соответствующему описателю Q, присваивается значение два, равное числу непереданных символов внешнего поля ввода.

7.2.1.14. Описатель : появляющийся в спецификации формата, вызывает завершение форматного управления при выполнении оператора ввода-вывода, если отсутствуют элементы в списке ввода-вывода. В противном случае описатель игнорируется.

Пример.

```
      TYPE 1, 3
      TYPE 2, 4,5
1     FORMAT ('__I=', I2:, 'J=', I2)
2     FORMAT ('__K=', I2:, 'L=', I2)
                  .
                  .
                  .
```
Данные операторы вызывают печать на терминал следующих строк:

I=__3

K=__4__L=__5

7.2.1.15. Описатель $ используется для подавления всякого продвижения при выполнении оператора ввода-вывода.

7.2.2. Разделители полей и записей. Разделители полей предназначены для разделения описателей полей и (или) групп описателей полей. Разделителями полей являются

дробная черта и запятая. Серия дробных черт также является разделителем полей. С помощью нескольких, идущих подряд, дробных черт можно осуществить пропуск вводимых записей или вывод пустых записей. Если между двумя описателями полей расположены N идущих подряд дробных черт, то это вызовет пропуск (N−1) записей при вводе или вывод (N−1) пустых записей. Однако, если N идущих подряд дробных черт встречаются до или после списка описателей полей, то они вызывают пропуск N записей при вводе или вывод N пустых записей, так как левая и правая скобки в спецификации формата являются соответственно признаками начала записи и ее завершения.

Если описатели полей в спецификации формата разделяются запятыми, то от первой левой скобки до последней правой скобки определяется одна запись.

Дробная черта используется не только для разделения описателей полей, но и для разграничения форматных записей.

Дробная черта выполняет роль признака завершения записи, вызывает завершение вводимой или выводимой записи и начало следующей записи.

Для разграничения внешних полей ввода при выполнении оператора ввода с терминала используется запятая.

Внешнее поле ввода, содержащее менее W символов, может быть завершено запятой, которая отменяет значение ширины поля, указанное в описателе IW, OW, FW.Y, EW.Y, DW.Y, GW.Y или LW.

Две идущие подряд запятые образуют пустое поле. В этом случае элемент списка ввода принимает значение нуль.

Завершение обработки одной записи на внешнем носителе не вызывает ввода или начала обработки следующей записи.

7.2.3. Взаимодействие форматного управления со списком ввода-вывода

Начало выполнения оператора форматного ввода или форматного вывода инициирует форматное управление. Каждое действие форматного управления определяется как очередным элементом списка ввода-вывода, если он имеется, так и очередным описателем поля в спецификации формата. Если имеется список ввода-вывода, то в спецификации формата должен существовать, по крайней мере, один описатель поля, отличный от NH, NX, '...', TN, :, $.

Если под форматным управлением выполняется оператор ввода, то при инициировании форматного управления читается одна запись; после этого следующие записи читаются

только в том случае, если этого требует спецификация формата. Каждое такое действие не может требовать большего числа символов, чем то, которое содержится в очередной записи.

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

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

Каждому описателю I, O, F, E, G, D, A, L, Q, интерпретируемому в спецификации формата, соответствует один элемент, задаваемый списком ввода-вывода, за исключением элемента типа комплексный, который требует интерпретации двух описателей F, E или G. Каждому описателю H, X, '...', T, :, $ не соответствует никакой элемент, задаваемый списком ввода-вывода; в этом случае форматное управление связывает информацию, содержащуюся в таком описателе, непосредственно с записью. Если встречается дробная черта, то спецификация формата требует начала новой и/или окончания очередной записи. Если при выполнении оператора ввода форматное управление завершается или встречается дробная черта, то все необработанные символы очередной записи пропускаются.

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

Если форматное управление дойдет до последней внешней правой скобки спецификации формата, то проверяется, остались ли необработанные элементы в списке ввода-вывода. Если таких элементов нет, то управление завершается; если же элементы в списке остались, то форматное управление требует начать новую запись, и управление возвращается к той спецификации повторений группы, которая заканчивается последней предшествующей правой скобкой, а при ее отсутствии — к первой левой скобке спецификации формата. Такое

действие, однако, не влияет на значение масштабного множителя.

7.2.4. Масштабный множитель

Предписатель масштабного множителя определен для использования с преобразованиями F, E, G, D и имеет вид NP где N — масштабный множитель — целая константа, значение которой должно лежать в диапазоне от −127 до +127.

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

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

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

Для E и D при выводе часть числа, образующая смешанную дробь (см. п. 4. 3.1.3), умножается на десять в степени N, а значение экспоненты умножается на десять в степени минус N.

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

Примеры:

Ввод

Описатель поля	Внешнее поле	Внутреннее значение
2PF7.4	—21.1—	0.211

Вывод

Описатель поля	Внутреннее значение	Внешнее поле
2PF5.1	2.3	230.0
3PE12.1	0.12543E+5	—125.4E+02
2PG10.3	0.01	10.000E−03
3PG10.5	1.0	1.0000—

7.2.5. Спецификация формата в массивах

Любой оператор форматного ввода-вывода или передачи данных может содержать имя массива в том месте, которое отведено для указателя формата. Если на этот массив производится ссылка из оператора, то начальная часть информации, содержащаяся в нем и взятая в естественном порядке, должна образовывать правильную спецификацию формата. На информацию, содержащуюся в массиве и следующую за правой скобкой, ограничивающей спецификацию формата, не накладывается никаких ограничений.

Спецификация формата, размещаемая в массиве, должна иметь тот же вид, что и в объявлении формата, т. е. она должна начинаться с левой скобки и заканчиваться правой скобкой. В массиве описатели полей вида NH и '...'. Не могут быть частью спецификации формата. Спецификация формата может быть помещена в массив с помощью объявления начальных данных (см. п. 8.2), а также с помощью оператора ввода с форматом, содержащим описатель поля AW.

Спецификация формата не может быть помещена в виртуальный массив.

Пример.

```
DIMENSION  A(5)
DATA A/' (X', '12, ', 'X, ', 'I3', ')'/
K=25
L=122
WRITE (7,A) K,L
        .
        .
        .
```

Спецификация формата помещена в массив A с помощью объявления начальных данных. Элементам массива A присваиваются следующие значения:

```
A(1)='(X'
A(2)='I2,'
A(3)='X,'
A(4)='I3'
A(5)=')'
```

по оператору форматного вывода выводятся переменные K и L следующим образом:

25—122

7.3. Основные операторы ввода-вывода

7.3.1. Операторы ввода-вывода последовательного доступа

7.3.1.1. Оператор форматного ввода имеет вид

READ (U,F[,END=S][,ERR=S])[K]

или

READ F[,K]

где U — номер устройства ввода;
F — указатель формата;
S — метка оператора;
K — список ввода.

В результате выполнения этого оператора вводятся очередные записи с устройства, заданного U. Вводимые записи просматриваются и преобразуются в соответствии с форматом, заданным F. Полученные в результате значения присваиваются элементам, определенным списком K.

Если в операторе отсутствует список ввода, тогда связанное с ним объявление формата или массив, содержащий спецификацию формата, должны содержать не менее одного описателя поля H или '....' для приема вводимых данных (см. п.п. 7.2.1.9, 7.2.1.10).

По оператору READ F[,K] очередные записи вводятся с системного устройства.

Примеры:

```
      READ (1,100)
100   FORMAT (F4.1)
```

Ввод записи с логического устройства 1. Значение, полученное после преобразования в соответствии с форматом 100, присваивается переменной X.

```
      READ 20, X,Y,Z
20    FORMAT (3F3.1)
```

Ввод записи с системного устройства. Значения, полученные после преобразования в соответствии с форматом 20, присваиваются переменным X,Y,Z.

7.3.1.2. Оператор форматного вывода имеет вид

WRITE (U,F[,ERR=S]) [K]

где U — номер устройства вывода;
F — указатель формата;
S — метка оператора;
K — список вывода.

В результате выполнения этого оператора создаются очередные записи на устройстве, заданном U. Список вывода K определяет последовательность передаваемых значений. Эти значения преобразуются и разносятся по позициям в соответствии с форматом, заданным F.

Если в операторе отсутствует список вывода, то форматное управление связывает информацию, содержащуюся в опи-

сателе H или '...' (см. п.п. 7.2.1.9, 7.2.1.10), непосредственно с создаваемой записью на устройстве, заданном U.

Примеры:

```
      WRITE (1,100) X,Y,Z
100   FORMAT (3F8.5)
```

Вывод записи, состоящей из трех полей, на логическое устройство 1.

```
      WRITE (2,200)
200   FORMAT ('MAIN')
```

Вывод буквенно-цифрового литерала 'MAIN' на логическое устройство 2.

7.3.1.3. Оператор форматного ввода с терминала имеет вид

ACCEPT F[,K]

где F — указатель формата;
K — список ввода.

Выполнение этого оператора идентично выполнению оператора форматного ввода (см. п. 7.2.1.1) за исключением того, что ввод осуществляется с конкретного логического устройства, как правило, с клавиатуры терминала.

Пример.

```
      ACCEPT 1,X
1     FORMAT (F3.1)
```

Ввод значения с клавиатуры терминала, которое после преобразования в соответствии с форматом 1 присваивается переменной X.

7.3.1.4. Оператор форматного вывода на терминал имеет вид

TYPE F[,K]

где F — указатель формата;
K — список вывода.

Выполнение этого оператора идентично выполнению оператора форматного вывода (см. п. 7.3.1.2) за исключением того, что вывод осуществляется на конкретное логическое устройство, это — терминал.

Пример.

```
      TYPE 1,X
1     FORMAT (F3.1)
```

Вывод значения переменной X на терминал в соответствии с форматом 1.

7.3.1.5. Оператор форматного вывода на построчно-печатающее устройство имеет вид

PRINT F[,K]

где F — указатель формата;
K — список вывода.

Выполнение этого оператора идентично выполнению оператора форматного вывода (см. п. 7.3.1.2) за исключением того, что вывод осуществляется на конкретное логическое устройство, как правило, на построчно-печатающее устройство.

Пример.

```
      PRINT 3
3     FORMAT ('END')
```

Вывод буквенно-цифрового литерала 'END' на построчно-печатающее устройство.

7.3.1.6. Оператор бесформатного ввода имеет вид

READ (U[,END=S] [,ERR=S]) [K]

где U — номер устройства ввода;
S — метка оператора;
K — список ввода.

В результате выполнения этого оператора с устройства, заданного U, вводится очередная запись и, если имеется список ввода, значения, содержащиеся в этой записи, последовательно без преобразования присваиваются элементам, определенным списком K. Последовательность значений, определяемая списком, не должна быть длиннее последовательности данных в бесформатной записи. Если вводимая запись содержит больше полей, чем имеется элементов в списке ввода, оставшаяся часть записи игнорируется. Если в записи содержится недостаточно полей, чтобы удовлетворить требование списка ввода, фиксируется состояние ошибки.

Если в операторе отсутствует список ввода, на устройстве осуществляется пропуск записи.

Примеры:

```
READ (1) X
```

Ввод записи с логического устройства 1. Значение присваивается переменной X.

```
READ (2)
```

Пропуск одной записи на логическом устройстве 2.

7.3.1.7. Оператор бесформатного вывода имеет вид

WRITE (U[,ERR=S]) [K]

где U — номер устройства вывода;
S — метка оператора;
K — список вывода

В результате выполнения этого оператора на устройстве,

заданном U, создается очередная запись, состоящая из последовательности значений, определяемой списком K; преобразования данных не происходит.

Если в операторе отсутствует список вывода, на устройстве создается пустая запись.

Примеры:.

WRITE (1) X

Вывод значения переменной X на логическое устяойство 1.

WRITE (2)

Вывод пустой записи на логическое устройство 2.

7.3.1.8. Оператор ввода с преобразованием по списку имеет вид

READ (U,∗[,END=S] [,ERR=S]) K

или

READ ∗,K

где U — номер устройства ввода;
∗ — указатель преобразования по списку;
S — метка оператора;
K — список ввода.

В результате выполнения этого оператора вводятся очередные записи с устройства, заданного U. Вводимые данные просматриваются и преобразуются в соответствии с типом элементов списка ввода (см. п. 6.1.1). Полученные в результате значения присваиваются элементам, определенным списком K.

По оператору READ∗K очередные записи вводятся с системного устройства.

Внешняя вводимая запись содержит последовательность элементов, которые представляют константы и разделители.

Константы могут быть типа целый, вещественный, двойной точности, комплексный или логический. Текстовые и восьмеричные константы не допускаются.

Константы записываются в стандартной форме за исключением логических констант, которые записываются в виде T для .TRUE. и F для .FALSE.. Комплексные константы допускается записывать в виде пары целых чисел.

Перед каждой константой может быть записана конструкция J∗, где J — ненулевая целая константа без знака. Наличие конструкции J∗ перед константой эквивалентно записи этой константы J раз через разделитель.

Разделителями являются:

— один или несколько пробелов (или символ ГТ);
— запятая с или без окружающих пробелов (или символов ГТ);

— дробная черта.

Наличие двух идущих подряд запятых эквивалентно записи нулевого значения.

Пример.

```
DOUBLE PRECISION A
COMPLEX B
LOGICAL C
READ (5,∗) I,D,A,B,C
      .
      .
      .
```

Внешняя вводимая запись содержит:

2.2—1.5—3.13,(1.,2.)—,—T/

элементам списка ввода I, D, A, B, C будут присвоены следующие значения: 2, 1.5, 3.13D 00, (1.0,2.0), .TRUE. соответственно.

7.3.1.9. Оператор вывода с преобразованием по списку имеет вид

WRITE (U,∗[,ERR=S]) K

где U — номер устройства вывода;
∗ — указатель преобразования по списку;
S — метка оператора;
K — список вывода.

В результате выполнения этого оператора создаются очередные записи на устройстве, заданном U. Список вывода K определяет последовательность передаваемых значений. Эти значения преобразуются и разносятся по позициям в соответствии с типом элементов списка вывода.

Внешнее поле вывода содержит данное, формат которого определяется в соответствии с типом элемента списка вывода (табл. 13).

Таблица 13

Тип данных	Формат
LOGICAL∗1	I5
LOGICAL∗4	L2
INTEGER∗2	I7
INTEGER∗4	I12
REAL∗4	1PG15.7
REAL∗8	1PG25.16
COMPLEX∗8	1X,'(',1PG14.7,',',1PG14.7,')'
Буквенно-цифровой литерал и константа Холлерита	1X,NA1 (где N — число символов)

Пример.

```
LOGICAL*1 L
INTEGER*2 A
REAL*4 C
L=.TRUE.
DATA A,C/5,2.2/
WRITE (7,*) 'ARRAY'
WRITE (7,*) L,A,C
      .
      .
      .
```

Внешние выводимые записи имеют вид

```
—ARRAY
————T ————5—2.200000 ——
```

7.3.1.10. Оператор ввода с терминала с преобразованием по списку имеет вид ACCEPT *,K

где * — указатель преобразования по списку;
K — список ввода.

Выполнение этого оператора идентично выполнению оператора ввода с преобразованием по списку (см. п. 7.3.1.8). За исключением того, что ввод осуществляется с конкретного логического устройства, как правило, с клавиатуры терминала.

Пример.

```
ACCEPT *,I,R
```

Внешняя вводимая запись содержит:

```
5.1.—1.2—/
```

Элементам списка ввода I, R будут присвоены значения 5 и 1.2 соответственно.

7.3.1.11. Оператор вывода на терминал с преобразованием по списку имеет вид TYPE *,K

где * — указатель преобразования по списку;
K — список вывода.

Выполнение этого оператора идентично выполнению оператора вывода с преобразованием по списку (см. п. 7.3.1.9) за исключением того, что вывод осуществляется на конкретное логическое устройство, это — терминал.

Пример.

```
DIMENSION A(3)
DATA A/3*1./
TYPE *,'KON'
TYPE *,(A(I),I=1,3)
```



```
      .
      .
      .
```

Внешние выводимые записи имеют вид

```
KON
—1.000000 ————1.000000 ————1.000000
```

7.3.1.12. Оператор вывода на построчно-печатающее устройство с преобразованием по списку имеет вид PRINT *,K

где * — указатель преобразования по списку;
K — список вывода.

Выполнение этого оператора идентично выполнению оператора вывода с преобразованием по списку (см. п. 7.3.1.9.) за исключением того, что вывод осуществляется на конкретное логическое устройство, как правило, построчно-печатающее устройство.

Пример.

```
I=2
A=—5.0
PRINT *, I,A
PRINT * 'PROGRAM'
      .
      .
      .
```

Внешние выводимые записи имеют вид

```
———2——-5.000000
PROGRAM
```

7.3.2. Операторы ввода-вывода прямого доступа осуществляют передачу двоичных записей фиксированной длины в произвольном порядке. Они используются при работе с файлами прямого доступа. Файл прямого доступа должен быть предварительно определен в операторе описания файлов (см. п. 7.4.4). При создании файлов прямого доступа первым оператором должен быть оператор бесформатного вывода прямого доступа.

Операторы ввода-вывода прямого доступа могут также использоваться при последовательной обработке файлов, если для этого следует использовать в качестве номера записи ассоциированную переменную.

7.3.2.1. Оператор бесформатного ввода прямого доступа имеет вид READ (U'R[,ERR=S]) [K]

где U — номер устройства ввода;
R — выражение типа целый, определяющее номер записи;
S — метка оператора;
K — список ввода.

Выполнение этого оператора идентично выполнению оператора бесформатного ввода (см. п. 7.3.1.6) за исключением того, что вводится запись, последовательный номер которой R.

Пример.

READ (1'3) X,Y,Z

Ввод записи с номером 3 с логического устройства 1. Значения присваиваются переменным X,Y,Z.

7.3.2.2. Оператор бесформатного вывода прямого доступа имеет вид: WRITE (U'R[,ERR=S]) [K]

где U — номер устройства вывода;
R — выражение типа целый, определяющее номер записи;
S — метка оператора;
K — список вывода.

Выполнение этого оператора идентично выполнению оператора бесформатного вывода (см. п. 7.3.1.7) за исключением того, что создается запись, последовательный номер которой R.

Если размер данных, которые должны быть переданы, превышает длину записи, то фиксируется состояние ошибки.

Пример.

WRITE (1'2) X

Вывод значения переменной X в запись с номером 2, находящуюся на логическом устройстве 1.

7.4. Вспомогательные операторы ввода-вывода

7.4.1. Оператор перемотки имеет вид REWIND U

где U — номер устройства.

В результате выполнения этого оператора устройство, заданное U, устанавливается в начало файла, открытого в данный момент времени на этом устройстве или в начальную позицию.

Оператор перемотки используется для проведения операций ввода-вывода последовательного доступа.

7.4.2. Оператор сдвига назад имеет вид BACKSPACE U

где U — номер устройства.

В результате выполнения этого оператора позиция устройства, заданного U, изменяется таким образом, что запись, которая до выполнения этого оператора была предыдущей, становится очередной; если же устройство находится в начальной позиции, то оператор не оказывает никакого действия.

Оператор сдвига назад используется для проведения операций ввода-вывода последовательного доступа.

7.4.3. Оператор разметки имеет вид ENDFILE U

где U — номер устройства.

В результате выполнения этого оператора на устройство, заданное U, выводится специальная запись «Конец файла». «Конец файла» — это единственная запись, обозначающая границу последовательного файла. Если запись «Конец файла» встретится при выполнении какого-либо оператора ввода, то действие такого оператора определяется наличием параметра END=S в операторе ввода; при отсутствии такого параметра фиксируется состояние ошибки. Оператор разметки используется для проведения операций ввода-вывода последовательного доступа.

7.4.4. Оператор описания файлов имеет вид

DEFINE FILE U1(M1, N1, U, I1) [,U2(M2, N2, U, I2)]...

где каждое UJ — константа или переменная типа целый, определяющая номер логического устройства, связанного с описываемым файлом;
каждое MJ — константа или переменная типа целый, определяющая число записей в файле (от 1 до MJ);
каждое NJ — константа или переменная типа целый, определяющая максимальную длину (в словах) записи в файле;
U — параметр указывает, что записи в файле являются бесформатными (двоичными), параметр U является единственным допустимым параметром в этой позиции;
каждое IJ — имя переменной типа целый, определяющее ассоциированную переменную.

Оператор DEFINE FILE задает характеристики файлов прямого доступа. Он указывает, что файл, состоящий из MJ записей по количеству слов не более NJ каждая, находится или должен находиться на логическом устройстве UJ. Все записи в файле пронумерованы последовательно от 1 до MJ. По окончании операции ввода-вывода ассоциированная переменная IJ принимает значение, равное номеру записи, которая непосредственно следует за последней переданной записью. Оператор описания файлов должен логически предшествовать операторам ввода-вывода прямого доступа READ, WRITE и FIND (см. п.п. 7.3.2.1, 7.3.2.2, 7.4.5), обрабатывающим файлы на указанном устройстве.

Пример.

DEFINE FILE 1(5,8,U,I)

Оператор определяет файл, находящийся на логическом устройстве 1 и состоящий из пяти записей (пронумерованных от 1 до 5) по 8 слов каждая. Все записи — бесформатные.

7.4.5. Оператор поиска записи имеет вид FIND (U'R)

где U — номер устройства;

R — выражение типа целый, определяющее номер записи ($1 <= R <= M$, где M — число записей в файле).

В результате выполнения этого оператора файл, связанный с устройством U, будет установлен на запись с номером R, и ассоциированная переменная примет значение, равное номеру этой записи. Никакой передачи данных не происходит.

Оператор поиска записи позволяет производить поиск следующей записи в то время, когда обрабатывается текущая запись, сокращая тем самым время выполнения программы.

Оператор поиска записи используется для проведения операций ввода-вывода прямого доступа.

7.4.6. Оператор открытия файла имеет вид

OPEN (P[, P] ...)

где P — ключевое слово и его значение, соединенные символом равно.

Перечень ключевых слов и их значений приведен в табл. 14.

Таблица 14

Ключевое слово	Наименование	Значение
1	2	3
UNIT	номер логического устройства	E
NAME	имя файла	N
TYPE	вид файла	'NEW', 'OLD' 'SCRATCH' 'UNKNOWN'
ACCESS	метод доступа	'SEQUENTIAL' 'DIRECT'
READONLY	чтение файла	—
FORM	вид записи	'FORMATTED' 'UNFORMATTED'
RECORDSIZE	длина записи	E
ERR	переход по ошибке	S
BUFFERCOUNT	число буферов	E
INITIALSIZE	число блоков	E

1	2	3
DISP[OSE]	распоряжение файлом	'SAVE' 'KEEP' 'DELETE'
ASSOCIATEVARIABLE	имя ассоциированной переменной	V
MAXREC	число записей	E

E — числовое выражение типа целый, вещественный или двойной точности. Результат выражения преобразуется к типу целый;

N — имя переменной, имя массива, имя элемента массива или буквенно-цифровой литерал;

S — метка оператора;

V — имя переменной типа целый

Оператор открытия файла связывает существующий файл (последовательного или прямого доступа) или вновь создаваемый с логическим устройством и задает характеристики файла.

Порядок записи ключевых слов в операторе произвольный. Если ключевое слово отсутствует, используется значение по умолчанию.

Ниже приводится описание ключевых слов.

Ключевое слово UNIT определяет номер логического устройства, которое должно быть связано с файлом. С логическим устройством может быть связан только один открытый файл.

Ключевое слово NAME определяет имя файла, связываемого с логическим устройством.

Имя файла, содержащееся в массиве, элементе массива или переменной должно ограничиваться нулевым байтом.

Имя файла не может содержаться в виртуальном массиве и элементе виртуального массива.

Ключевое слово TYPE определяет вид открываемого файла.

Оно может принимать одно из следующих значений: 'OLD', 'NEW', SCRATCH', 'UNKNOWN'.

'OLD' определяет существующий файл.

'NEW' определяет новый файл.

'SCRATCH' определяет временный файл.

'UNKNOWN' определяет как 'NEW', если файл с указанным именем на устройстве не существует, в противном случае — как 'OLD'. По умолчанию используется значение 'NEW'.

Ключевое слово ACCESS определяет метод доступа к файлу: прямой или последовательный. Оно может принимать одно из следующих значений: 'DIRECT', 'SEQUENTIAL'.
'DIRECT' определяет файл прямого доступа.
'SEQUENTIAL' определяет файл последовательного доступа.

По умолчанию используется значение 'SEQUENTIAL'.

Ключевое слово READONLY определяет, что файл предназначен только для чтения. Запись в этот файл запрещена.

Ключевое слово FORM определяет вид записи. Оно может принимать одно из следующих значений: 'FORMATTED', 'UNFORMATTED'.
'FORMATTED' определяет форматную запись.
'UNFORMATTED' определяет бесформатную запись.

По умолчанию используется значение 'FORMATTED' для файлов последовательного доступа и 'UNFORMATTED' — для файлов прямого доступа.

Ключевое слово RECORDSIZE определяет максимальную длину записи файла прямого доступа. Длина записи указывается в двойных словах. Оптимальная длина — число, кратное 256, или его делитель.

Ключевое слово ERR определяет оператор, помеченный меткой S, которому передается управление при возникновении состояния ошибки.
Состоянием ошибки является:
— синтаксическая ошибка в имени файла;
— файл не найден;
— файл уже открыт на устройстве;
— файл не может быть открыт на устройстве.
При обнаружении состояния ошибки файл не открывается.

Ключевое слово ERR действительно только для данного открытого файла.

Ключевое слово BUFFERCOUNT определяет число буферов для обслуживания данного устройства ввода-вывода.

По умолчанию используется значение, равное единице.

Ключевое слово INITIALSIZE определяет число блоков на диске для вновь создаваемого файла.

По умолчанию используется значение, равное числу блоков в наибольшей из двух областей:
— половина наибольшей свободной области на томе;
— вторая по величине свободная область на томе.

Ключевое слово DISP[OSE] определяет, как следует распорядиться файлом после его закрытия. Оно может принимать одно из следующих значений: 'SAVE', 'KEEP', 'DELETE'.
'SAVE' — файл сохраняется.

'KEEP' — то же, что и 'SAVE'.
'DELETE' — файл удаляется после закрытия. Файл, предназначенный для чтения, не может быть удален.

По умолчанию используется значение 'SAVE'.

Ключевое слово ASSOCIATEVARIABLE определяет целую переменную, которая по окончании операции ввода-вывода принимает значение номера записи, непосредственно следующей за последней переданной записью. Это ключевое слово игнорируется для файлов последовательного доступа.

Ключевое слово MAXREC определяет максимальное число записей в файле прямого доступа. Для файла последовательного доступа это ключевое слово игнорируется.

По умолчанию для вновь открываемого файла используется наибольшее из двух значений:
— INITIALSIZE;
— произведение MAXREC и RECORDSIZE.

Пример.

```
OPEN (UNIT=1, TYPE='NEW', ACCESS='SEQUENTIAL',
DISP='KEEP')
```

Открывается новый файл последовательного доступа на логическом устройстве 1, после закрытия он сохраняется.

7.4.7. Оператор закрытия файла имеет вид

```
CLOSE (UNIT=U [,(DISP[OSE]=P] [,ERR=S])
```

где U — номер устройства;
P — одно из следующих значений: 'SAVE', 'KEEP', 'DELETE';
S — метка оператора.

Ключевые слова DISP[OSE], ERR и их значения определяются так же, как и в операторе открытия файла (см. п. 7.4.6).

Оператор закрытия файла устраняет связь между файлом и устройством, подтверждает или переопределяет указание о распоряжении файлом, данное в операторе открытия файла.

Пример.

```
CLOSE (UNIT=2, DISP='DELETE')
```

Закрывается файл на логическом устройстве 2 и стирается.

7.5. Операторы передачи аналогичны по действию операторам форматного ввода-вывода за исключением того, что передача данных осуществляется из одной области оперативной памяти в другую без использования устройств ввода-вывода.

7.5.1. Оператор передачи с преобразованием данных в код КОИ-7 имеет вид ENCODE (C, F, V [, ERR=S]) [K]

где C — выражение типа целый, определяющее число преоб-

разуемых символов;
F — указатель формата;
V — имя массива, элемента массива или переменной;
S — метка оператора;
K — список вывода.

Этот оператор преобразует данные из внутреннего представления в код КОИ-7 в соответствии со спецификацией формата. Преобразованные данные запоминаются в V.

Оператор ENCODE аналогичен оператору форматного вывода последовательного доступа (см. п. 7.3.1.2.).

Пример.

```
DIMENSION A(3), K(3)
DATA K/1234, 5678, 9012/
     ENCODE (12, 1, A) K
1    FORMAT (314)
      .
      .
      .
```

Данные массива K из внутреннего представления преобразуются в код КОИ-7 в соответствии со спецификацией формата и запоминаются в массиве A:

A(1) = '1234'
A(2) = '5678'
A(3) = '9012'

7.5.2. Оператор передачи с преобразованием данных во внутреннее представление имеет вид

DECODE (C, F, V [, ERR=S]) [K]

где C — выражение типа целый, определяющее число преобразуемых символов;
F — указатель формата;
V — имя массива, элемента массива или переменной;
S — метка оператора;
K — список ввода.

Этот оператор преобразует данные из кода КОИ-7 во внутреннее представление в соответствии со спецификацией формата и записывает преобразованные данные в список ввода. V содержит символы, которые должны быть преобразованы.

Оператор DECODE аналогичен оператору форматного ввода последовательного доступа (см. п. 7.3.1.1).

Пример.

```
DIMENSION A(3), K(3)
DATA A/'1234', '5678', '9012'/
DECODE (12, 1, A) K
1    FORMAT (3I4)
      .
      .
      .
```

Данные массива A из символьного формата преобразуются во внутреннее представление в соответствии со спецификацией формата и записываются в массив K:

K(1) = 1234
K(2) = 5678
K(3) = 9012

7.6. Вывод форматных записей на печать

При передаче форматной записи на печать ее первый символ не печатается, а определяет продвижение по вертикали (табл. 15).

Таблица 15

Символ	Продвижение по вертикали перед печатью
1	2
пробел	одна строка
ноль	две строки
1	к первой строке следующей страницы
+	никакого продвижения
$	перед печатью — продвижение по вертикали на одну строку, по окончании печати — подавление продвижения

Символ, определяющий продвижение по вертикали перед печатью, задается в форме 1H или '...'. Символ $ может задаваться как в виде 1H$, '$', так и $.

8. ОБЪЯВЛЕНИЯ

Объявления описывают способ использования программы, редактирования и размещения данных, а также характеристики операндов и вводимые в употребление функции.

Имеются пять типов объявлений:
- объявления спецификаций;
- обявления начальных данных;
- объявление формата;
- объявление внутренней функции;
- заголовки (функций, подпрограмм, спецификаций блоков данных, головного модуля).

Объявления не должны быть помечены, кроме объявления формата.
Объявление формата описано в п. 7.2.
Объявление внутренней функции и заголовки описываются в разделе 9.

8.1. Объявления спецификаций

Имеются пять типов объявлений спецификаций:
— объявление массивов;
— объявление общих объектов;
— объявление эквивалентности;
—объявление внешних имен;
— объявление типа.

8.1.1. Описание массива задает характеристики массива: указывает символическое имя массива, число измерений и размеры по каждому измерению. Описание массива может встречаться в объявлениях типа, массивов или общих объектов.

Описание массива имеет вид
V(I)
где V — символическое имя;
I — список границ.

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

Число выражений, образующих список границ, указывает число измерений массива. Значения выражений в описании массива определяют максимальное значение, которое может принимать индекс в любом имени элемента этого массива. Значение индекса в имени элемента массива не должно быть меньше единицы или больше максимального значения, определенного описанием этого массива.

8.1.1.1. Функция линеаризации массива и значение индекса

В табл. 16 Для заданных измерений списков границ и индексов в именах элементов массива приведены значения индексов этих элементов массива и максимальные значения, которые могут принимать индексы в именах элементов этих массивов. Значения всех индексных выражений должны быть больше нуля.

Функция линеаризации упорядочивает все элементы любого массива. Значение этой функции для некоторого данного

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

Таблица 16

Число измерений	Список границ	Индекс	Значение индекса	Максимальное значение индекса
1	2	3	4	5
1	(A)	(X)	X	A
2	(A, B)	(X, Y)	X+A∗(Y—1)	A∗B
3	(A, B, C)	(X, Y, Z)	X+A∗(Y—1)+B∗B∗(Z—1)	A∗B∗C

ПРИМЕЧАНИЕ. X, Y и Z — индексы выражения; A. B, C — верхние границы по измерениям.

8.1.1.2. Регулируемые размеры

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

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

8.1.2. Описание виртуального массива имеет такой же вид, что и описание обычного массива. Требования к списку границ для обычного массива справедливы и для виртуального массива.

Описание виртуального массива может встречаться в объявлении виртуальных массивов, имя виртуального массива — в объявлении типа.

Имя виртуального массива и элементы виртуального массива не могут использоваться:
— в объявлениях общих объектов, эквивалентности, начальных данных;
— как указатель формата в операторах ввода-вывода и передачи данных;
— в качестве третьего параметра в скобках в операторах передачи данных.

8.1.3. Объявление массивов имеет вид
DIMENSION V1(I1)[, V2(I2)]...
———————
где каждое VJ(IJ) — описание массива.

Объявление массивов определяет число измерений в массивах и размеры по каждому измерению.

Пример.
DIMENSION A(5, 7, 10, 13, 41), D(100)
объявление массивов определяет, что массив A имеет пять измерений и верхние границы по измерениям — 5, 7, 10, 13, 41, массив D имеет одно измерение, и верхняя граница его равна 100.

8.1.4. Объявление виртуальных массивов имеет вид
VIRTUAL V1(I1)[, V2(I2)]...
———————
где каждое VJ(IJ) — описание виртуального массива.

Пример.
VIRTUAL A(100)
объявление определяет, что виртуальный массив A имеет одно измерение, и верхняя граница его равна 100.

8.1.5. Объявление общих объектов имеет вид
COMMON [/[X1]/]A1 [[,]/[X2]/A2]...
———————
где каждое XI — символическое имя либо пусто;
каждое AI — непустой список имен переменных, имен массивов или описаний массивов.

В списки AI не должны входить формальные параметры, имена виртуальных массивов и имена элементов виртуальных массивов.

Если X1 пусто, то первые две дробные черты необязатель-

ны. Каждое XI, называемое именем блока, не имеет никакого отношения к каким-либо переменным или массивам с тем же самым символическим именем как в том программном модуле, в котором встречается это объявление общих объектов, так и в любом другом.

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

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

Размер общего блока в программном модуле равен сумме объемов памяти, которые требуются для размещения объектов, включенных в общий блок при помощи объявлений общих объектов и эквивалентности (п. 8.1.6). Размеры общих блоков с одним и тем же именем во всех программных модулях, входящих в выполняемую программу, должны совпадать. Однако размер непомеченного общего блока не обязан быть одним и тем же в разных совместно выполняемых программных модулях.

Ввиду того, что приписывание объектов общему блоку выполняется по принципу один к одному, то объекты, приписываемые объявлением общих объектов в одном программном модуле, должны соответствовать по типу данных объектам, помещаемым в общий блок в другом программном модуле.

Для общих блоков с одним и тем же именем или для непомеченного общего блока:
1) во всех тех программных модулях, где в данной позиции (определяемой числом предшествующих единиц памяти) за-

дан один и тот же тип, ссылки на эту позицию дают одно и то же значение;
2) правильная ссылка, сделанная на конкретную позицию, подразумевает тип, заданный в объявлениях, если последнее по времени присваивание в эту позицию было того же типа.

Пример.

Головной модуль
```
COMMON /A/I,E
      .
      .
      .
CALL S
      .
      .
      .
```
Подпрограмма
```
SUBROUTINE S
COMMON /A/J,D
      .
      .
      .
RETURN
END
```

В головном модуле объявление общих объектов помещает объекты I и E в общий блок с именем A. В подпрограмме S объявление общих объектов устанавливает соответствие между J и D, с одной стороны, и I и E, с другой, в общем блоке — с именем A.

8.1.6. Объявление эквивалентности имеет вид
EQUIVALENCE (K1)[, (K2)]...

где каждое KI — список вида A1, A2, ..., AMI;
каждое AJ — имя переменной, имя элемента массива, индекс которого содержит только имя массива, константы;
MI>=2.

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

Все элементы, образующие какой-либо список KI в Объявлении эквивалентности, размещаются в памяти, начиная с одной и той же единицы памяти. Объявление эквивалентности

не должно использоваться для того, чтобы сделать две или более величины математически эквивалентными.

Отведение памяти для переменных или массивов, объявленных непосредственно в объявлении общих объектов, производится только с учетом их типов, объявлений общих объектов и описаний массивов. Объявленным таким образом объектам всегда отводится память в том порядке, в каком они следуют в объявлении общих объектов.

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

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

Не допускается явное или неявное отведение одной и той же единицы памяти для хранения более чем одного элемента одного и того же массива.

При установлении эквивалентности между двумя элементами разных массивов объявление эквивалентности также устанавливает эквивалентность между соответствующими элементами, примыкающими к тем, которые названы в объявлении.

При использовании в объявлении эквивалентности элементов массива типа BYTE или LOGICAL*1 с данными других типов могут быть указаны только те элементы, которые размещены на границе слова, т. е. элементы с нечетными индексами.

Пример.
```
EQUIVALENCE (A, B, C)
```
объявление эквивалентности устанавливает эквивалентность между переменными A, B и C.

8.1.7. Объявление внешних имен имеет вид
EXTERNAL V1[, V2]...

где VI — имя внешней процедуры.

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

Пример.
Программный модуль
EXTERNAL SIN
.
.
.
CALL S(O:, SIN, YI)
.
.
.
Подпрограмма
SUBROUTINE S(X, F, Y)
Y=F(X)
RETURN
END

В результате выполнения подпрограммы получаем YI=SIN(0.)

8.1.8. Объявление типа имеет вид T V1[, V2]...
где T — одна из предусмотренных записей типа данных (табл. 17);

Каждое VI — имя переменной, имя массива, имя виртуального массива, имя функции или описание массива.

За T и каждым VI может следовать необязательный указатель длины в виде ∗N, где N — допустимая для данного типа длина в байтах. Запись VI∗N переопределяет для конкретного VI указатель длины, предусмотренный в записи T[∗N].

Объявление типа изменяет или подтверждает указание типа, определенное по IMPLICIT или по умолчанию.

Таблица 17

Тип данных	T		N
	без указателя длины	с указателем длины	
1	2		3
Целый	INTEGER	INTEGER∗2 INTEGER∗4	2 или 4 2 4
Вещественный	REAL	REAL∗4	4

1	2		3
Двойной точности	DOUBLE PRECISION	REAL∗8	8
Комплексный	COMPLEX	COMPLEX∗8	8
Логический	LOGICAL	LOGICAL∗4 LOGICAL∗1	4 1
Байтовый	BYTE		1

Пример.
INTEGER∗2 S, A∗4

Переменные S и A являются переменными типа целый, в памяти они занимают 2 и 4 байта соответственно.

8.1.8.1. Правила, определяющие запись текстовых данных в памяти

Если в выражении используется буквенно-цифровой литерал или константа Холлерита, то запись их в памяти регулируется следующими правилами:

— в операторе присваивания запись текстовой константы в памяти определяется типом данного, которому присваивается эта константа.

Например:

Оператор	Тип данных	Длина константы
REL='ABCD'	REAL∗4	4
IF(I.EQ.'XY') GOTO 3	INTEGER∗2	2
M=N—'ABC'	INTEGER∗2	2
X='Z'	REAL∗4	4

— в контекстах, где используется определенный тип данных (обычно тип целый), тип этих данных определяет запись текстовой константы. Например, при записи элемента массива:

Y(IX)=Y('ABC')+3 INTEGER∗4 4

— если текстовая константа используется в качестве фактического параметра, то запись константы не определяется другими типами данных.

Например:
CALL APAC ('ABCDEFGHI') 9

— во всех других случаях предполагается запись по типу INTEGER∗2.

Например:

IF ('AB') 1, 2, 3	INTEGER*2	2
I='C'—'A'	INTEGER*2	2
J=.NOT.'B'	INTEGER*2	2

Если длина текстовой константы меньше длины, определяемой типом данных, то константа дополняется справа пробелами. Если длина текстовой константы больше длины, определяемой типом данных, то константа усекается справа.

8.1.9. Неявное объявление типа имеет вид

IMPLICIT T(A1[, A2]...) [, T(B1[, B2]...)]...

где T — одна из предусмотренных записей типа данных (см. табл. 17);

Каждое AI, BI — записывается в виде одной буквы алфавита или двух букв алфавита, разделенных символом «—».

Появление в IMPLICIT одной буквы AI или двух букв, разделенных символом «—» означает, что все вхождения в программный модуль имен (переменных, массивов, функций), начинающихся с указанной буквы или букв указанного диапазона алфавита, связываются с конкретным типом данных T.

Неявное объявление типа должно предшествовать объявлению типа.

Пример.

IMPLICIT REAL (A—F)

Имена переменных, массивов, функций, начинающиеся с букв A, B, C, D, E, F, связываются с типом вещественный.

8.2. Объявление начальных данных имеет вид

DATA K1/D1/[[,]K2/D2/]...

где каждое KI — список, содержащий имена переменных, массивов, элементов массивов;
каждое DI — список констант, перед каждой из которых может быть записана конструкция J*;
J — отличное от нуля целое без знака.

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

Формальные параметры, имена виртуальных массивов и элементы виртуальных массивов не могут входить в списки K. Любое индексное выражение должно быть целым без знака.

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

Объявление начальных данных используется для придания начальных значений некоторым переменным и элементам

массивов. В каждой паре этих списков должно существовать однозначное соответствие между именами списка KI с одной стороны и константами списка DI — с другой; начальные значения устанавливаются на основе этого соответствия. Переменная или элемент массива, которым придаются начальные значения, не могут входить в непомеченный общий блок. Переменной или элементу массива, входящим в помеченный общий блок, начальные значения могут быть приданы только в модуле-блоке данных.

Если в списке K встречается имя массива, то выполняется присваивание значений каждому элементу этого массива. Таким образом, соответствующий список D должен содержать достаточное количество констант, чтобы заполнить весь массив.

Объявление начальных данных должно следовать за всеми остальными объявлениями (если они есть в программном модуле).

Пример.

DATA A, B, C/2*3.1, 1.44/

Переменным A, B и C придаются следующие начальные значения:

A=3.1
B=3.1
C=1.44

9. ПРОЦЕДУРЫ И МОДУЛИ

В языке ФОРТРАН имеются три категории процедур: внутренние функции, внешние функции и внешние подпрограммы. Первые две категории процедур относятся к функциям (или процедурам-функциям), а последняя категория — к подпрограммам (или процедурам-подпрограммам).

Имеются две категории модулей: модули-процедуры и модули-спецификации. К первой из них относятся модули-функции и модули-подпрограммы. Модуль-процедура состоит из заголовка функции (п. 9.2.1) или заголовка подпрограммы (п. 9.3.1), за которым следует совокупность предложений языка ФОРТРАН с заключительной строкой после нее. Ко второй категории относятся модули-блоки данных. Модуль-блок данных состоит из заголовка спецификации блока данных (п. 9.4), за которым следует набор соответствующих объявлений начальных данных с заключительной строкой после него.

Все перечисленные категории процедур не должны обра-

щаться к себе — прямо или косвенно (через другие процедуры).

Одни и те же формальные параметры могут использоваться более чем в одной процедуре, их имена могут определять различные объекты программы в других предложениях программного модуля. В пределах одной процедуры формальные параметры не должны повторяться.

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

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

9.1.1. Структура объявлений внутренних функций имеет вид

F([A1[, A2]...]) = E

где F — символическое имя определяемой функции;
E — арифметическое или логическое выражение;
каждое AI — символическое имя, называемое формальным параметром.

Соответствие между F и E должно удовлетворять правилам присваивания, приведенным в п.п. 6.1.1 — 6.1.2.

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

Выражение E, кроме формальных параметров, может содержать только:
— нетекстовые константы;
— указатели переменных;
— указатели внутренних функций, определенных ранее в данном программном модуле;
— указатели внешних функций.

Примеры:

RAG (A, B, C, D) = (A+B+C)/D

VALUE(P) = 5.35*P**3

9.1.2. Ссылки на внутренние функции. Для ссылки на внутреннюю функцию используется ее указатель (п. 4.4) в качестве первичного выражения в арифметическом или логическом выражении. Фактические параметры, образующие список фактических параметров в указателе функции, должны согласовываться по порядку, числу и типу с соответствующими формальными параметрами. Фактическим параметром в указателе внутренней функции может быть любое выражение того же типа, что и соответствующий формальный параметр.

При появлении в выражении имени внутренней функции происходит замена формальных параметров значениями соответствующих фактических параметров и вычисление значения внутренней функции согласно объявлению. После этого полученное значение используется для вычисления выражения, содержащего обращение к внутренней функции.

9.2. Внешние функции. Внешняя функция определяется вне программного модуля, в котором есть на нее ссылка. Для определения внешней функции служит отдельный программный модуль, называемый модулем-функцией. Первым предложением этого модуля является заголовок функции.

9.2.1. Структура модулей-функций. Заголовок внешней функции имеет вид

[T] FUNCTION F[*N] [(A1[, A2]...])]

где T — одна из предусмотренных записей типа данных (см. табл. 17);
F — символическое имя определяемой функции;
*N — указатель длины данных;
каждое AI — символическое имя, называемое формальным параметром.

Заголовок внешней функции не должен иметь метку.

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

Символическое имя определяемой функции должно встречаться в модуле в качестве имени переменной. При каждом выполнении модуля эта переменная должна быть определена, причем впоследствии на эту переменную можно ссылаться или переопределять ее. Значение этой переменной к моменту выполнения в данном модуле любого оператора возврата называется значением определяемой функции. Символическое имя определяемой функции не должно встречаться

ни в одном объявлении этого программного модуля, за исключением вхождения в заголовок функции в качестве ее имени. Формальные параметры не могут встречаться в объявлениях эквивалентности, в общих объектах или начальных данных в этом модуле-функции. Модуль-функция при своем выполнении может определить или переопределить один или несколько своих фактических параметров, что, наряду с вычислением значения функции, также является результатом выполнения модуля-функции. Модуль-функция может содержать любые предложения, за исключением заголовка блока данных, заголовка подпрограммы, другого заголовка функции, а также любого предложения, которое явно или неявно ссылается на определяемую функцию. Модуль-функция должен содержать, по крайней мере, один оператор возврата.

Пример.

```
      FUNCTION ROOT (B)
      X=1.0
3     EX=EXP(X)
      EPIC=1./EX
      ROOT=((EX+EPIC)*.5+COS(X)—B)/((EX—EPIC)*.5—SIN(X))
      IF (ABS(X—ROOT).LT.1E—6) RETURN
      X=ROOT
      GO TO 3
      END
```

модуль-функция вычисляет корень уравнения

$$F(X)=\mathrm{COSH}(X)+\cos(X)-B=0$$

по формуле

$$X(I+1)=X(I)-[\mathrm{COSH}(XI)+\cos(XI)-B]/[\mathrm{SINH}(XI)-\sin(XI)]$$

вычисление повторяется до тех пор, пока разность между $X(I)$ и $X(I+1)$ не станет меньше, чем $10**(-6)$.

модуль-функция использует основные внешние функции — EXP, SIN, COS и ABS.

9.2.2. Ссылки на внешние функции. Для ссылки на внешнюю функцию используется ее указатель (п. 4.4) в качестве первичного выражения в арифметическом или логическом выражении. Фактические параметры, образующие список фактических параметров в указателе функции, должны согласовываться по порядку, числу и типу с соответствующими формальными параметрами модуля-функции. Фактическим параметром в указателе внешней функции может быть:

— имя переменной;
— имя элемента массива;
— имя массива;
— любое другое выражение;
— имя внешней процедуры.

Если фактический параметр является именем внешней

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

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

При появлении в выражении имени внешней функции происходит замена формальных параметров значениями соответствующих фактических параметров. Используя эти значения, выполняются предложения модуля-функции. Перед выполнением оператора возврата определяемой функции присваивается вычисленное значение. После того, как управление вернется в вызывающий программный модуль, это значение используется для вычисления выражения, содержащего обращение к внешней функции.

9.2.3. Основные внешние функции. Система ФОРТРАН снабжена основными внешними (библиотечными) функциями. Перечень основных внешних функций приведен в приложении 3. Ссылки на эти функции производятся так же, как описано в п. 9.2.2. Фактические параметры, для которых значения этих функций математически не определены или имеют тип, отличный от приведенного в приложении 3, являются недопустимыми.

9.3. Подпрограммы. Внешняя подпрограмма определяется вне того программного модуля, который ссылается на нее. Для определения внешней подпрограммы служит отдельный программный модуль, называемый модулем-подпрограммой. Первым предложением этого модуля является заголовок подпрограммы.

9.3.1. Структура модулей-подпрограмм. Заголовок подпрограммы имеет вид

SUBROUTINE S [([A1[, A2]...])]

где S — символическое имя определяемой подпрограммы;
каждое AI — символическое имя, называемое формальным параметром.

Заголовок подпрограммы не должен иметь метку.

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

Символическое имя определяемой подпрограммы не дол-

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

Пример.

```
      Головной модуль
      DO 5 I=1, 25
      CALL KVADR (I)
      WRITE (7, 2) J
5     CONTINUE
2     FORMAT (I5)
      STOP
      END
      Подпрограмма
      SUBROUTINE KVADR (K)
      J=K**2
      RETURN
      END
```

В головном модуле выводятся на терминал квадраты чисел от 1 до 25, которые вычисляются в подпрограмме.

9.3.2. Ссылки на внешние подпрограммы. Для ссылки на внешнюю подпрограмму используется оператор вызова подпрограммы (п. 6.2.4). Фактические параметры, образующие список фактических параметров этого оператора, должны согласовываться по порядку, числу и типу с соответствующими формальными параметрами модуля-подпрограммы. Исключением из правил согласования типов является использование в качестве фактического параметра текстовой константы.

Фактическим параметром при ссылке на внешнюю подпрограмму может быть:
— текстовая константа;
— имя переменной;
— имя элемента массива;

— любое другое выражение;
— имя внешней процедуры.

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

Если фактический параметр соответствует формальному параметру, который в подпрограмме определяется либо переопределяется, то фактический параметр должен быть либо именем переменной, либо именем элемента массива, либо именем массива.

Возврат в вызывающий программный модуль осуществляется по оператору возврата (п. 6.2.5).

9.4. Модуль-блок данных. Заголовок спецификации блока данных имеет вид BLOCK DATA [S]

где S — символическое имя блока данных.

Заголовок спецификации блока данных не должен иметь метку.

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

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

Пример.

```
      BLOCK DATA
      INTEGER X, Y
      LOGICAL Q, W
      DOUBLE PRECISION D
      DIMENSION R(4)
      COMMON /BLK1/R, X, Q, D/BLK2/W, Y, Z
      DATA R/2.0, 3*4.01/ ,Q/ .FALSE. / ,D/0.317638D—7 / ,
1     W/ .TRUE. / ,Z/3.5/
      END
```

Модуль-блок данных служит для придания начальных значений переменным X, Q, D и массиву R из общего блока с именем BLK1 и переменным W,Y, Z — из общего блока с именем BLK2.

9.5. Головной модуль. Для определения головного модуля средствами системы ФОРТРАН служит отдельный программный модуль, первым предложением которого является заголовок головного модуля.

Заголовок головного модуля имеет вид

PROGRAM S

где S — символическое имя головного модуля.

Заголовок головного модуля не должен иметь метку.

Символическое имя S не должно совпадать ни с каким именем любого другого объекта внутри головного модуля, именем подпрограммы, именем внешней функции, именем спецификации блока данных, именем блока в объявлениях общих объектов.

Пример.

PROGRAM NAME

головному модулю придается имя NAME

10. СРЕДСТВА ОТЛАДКИ ПРОГРАММЫ

Отладочными средствами языка ФОРТРАН являются:
— строки отладки;
— оператор PAUSE;
— диагностические сообщения;
— программа «Диалоговый отладчик».

Строки отладки (п. 3.3.2). Можно включать на ранних этапах разработки программы, а позднее рассматривать их как обычные комментарии.

Использование оператора PAUSE с выводом на терминал сообщений (п. 6.2.8) обеспечивает пользователя методом контроля работы программы. Строки с оператором PAUSE могут включаться в программу на этапе отладки и исключаться из окончательного варианта программы.

Диагностические сообщения выдаются на этапах трансляции и выполнения программы. Сообщения и рекомендации по их устранению приведены в документах [2] и [3].

Программа «Диалоговый отладчик» (см. [4]) обеспечивает отладку программ на языке ФОРТРАН в интерактивном режиме.

ПРИЛОЖЕНИЕ 1

ОПЕРАЦИИ ФОРТРАН

В данном приложении приведены операции языка ФОРТРАН (см. таблицу).

Таблица

Тип операции	Знак операции	Операция
1	2	3
Арифметический	**	Возведение в степень
	*, /	Умножение, деление
	+, —	Сложение, вычитание
Отношения	.GT.	Больше
	.GE.	Больше или равно
	.LT.	Меньше
	.LE.	Меньше или равно
	.EQ.	Равно
	.NE.	Не равно
Логический	.NOT.	Логическое отрицание
	.AND.	Логическое сложение
	.OR.	Логическое умножение
	.EQV.	Логическая эквивалентность
	.XOR.	Исключающее ИЛИ

ПРИМЕЧАНИЕ. Арифметические и логические операции расположены в порядке убывания приоритета.

ПРИЛОЖЕНИЕ 2

ПРЕДЛОЖЕНИЯ ЯЗЫКА ФОРТРАН

В данном приложении приведены предложения языка ФОРТРАН (см. таблицу).

Таблица

Формат предложения	Пункт
1	2
ОПЕРАТОРЫ ПРИСВАИВАНИЯ	
V=E	6.1.1
V=E	6.1.2
ASSIGN K TO I	6.1.3

1	2
ОПЕРАТОРЫ УПРАВЛЕНИЯ	
GO TO K	6.2.1.1
GO TO (K1[,K2]...)[,] I	6.2.1.3
GO TO I [[,](K1[,K2]...)]	6.2.1.2
IF(E) K1,K2,K3	6.2.2
IF(E) S	6.2.3
DO N [,] I=M1,M2[,M3]	6.2.6
CONTINUE	6.2.7
CALL S [([A1][,[A2]]...)]	6.2.4
RETURN	6.2.5
PAUSE [N]	6.2.8
STOP [N]	6.2.9
ОПЕРАТОРЫ ВВОДА-ВЫВОДА ПОСЛЕДОВАТЕЛЬНОГО ДОСТУПА	
READ (U[,END=S][,ERR=S]) [K]	7.3.1.6
WRITE (U[,ERR=S]) [K]	7.3.1.7
READ (U,F[,END=S][,ERR=S]) [K]	7.3.1.1
и	
READ F[,K]	
WRITE (U,F[,ERR=S]) [K]	7.3.1.2
ACCEPT F[,K]	7.3.1.3
TYPE F[,K]	7.3.1.4
PRINT F[,K]	7.3.1.5
ОПЕРАТОРЫ ВВОДА-ВЫВОДА С ПРЕОБРАЗОВАНИЕМ ПО СПИСКУ	
READ (U,*[,END=S][,ERR=S] K	7.3.1.8
и	
READ *,K	
WRITE (U,*[,ERR=S]) K	7.3.1.9
ACCEPT *,K	7.3.1.10
TYPE *,K	7.3.1.11
PRINT *,K	7.3.1.12
ОПЕРАТОРЫ ВВОДА-ВЫВОДА ПРЯМОГО ДОСТУПА	
READ (U'R[,ERR=S]) [K]	7.3.2.1
WRITE (U'R[,ERR=S]) [K]	7.3.2.2
ВСПОМОГАТЕЛЬНЫЕ ОПЕРАТОРЫ ВВОДА-ВЫВОДА	
DEFINE FILE U1(M1,N1,U,I1)[,U2(M2,N2,U,I2)]...	7.4.4
REWIND U	7.4.1
BACKSPACE U	7.4.2
ENDFILE U	7.4.3
FIND (U'R)	7.4.5
OPEN (P[,P]...)	7.4.6
CLOSE (P[,P]...)	7.4.7
ОПЕРАТОРЫ ПЕРЕДАЧИ	
ENCODE (C,F,V[,ERR=S]) [K]	7.5.1
DECODE (C,F,V[,ERR=S]) [K]	7.5.2
ОБЪЯВЛЕНИЯ	
FORMAT (Q1T1Z1T2Z2...ZN—1TNQ2)	7.2
IMPLICIT T(A1[,A2]...)[,T(B1[,B2]...)]...	8.1.9
T V1[,V2]...	8.1.8
DIMENSION V1(I1)[,V2(I2)]...	8.1.3
VIRTUAL V1(I1)[,V2(I2)]...	8.1.4

1	2
COMMON [/[X1]/]A1[[,]/[X2]/A2]...	8.1.5
EQUIVALENCE (K1)[,(K2)]...	8.1.6
EXTERNAL V1[,V2]...	8.1.7
DATA K1/D1/[[,] K2/D2/]...	8.2
F([A1[,A2]...])=E	9.1.1
[T] FUNCTION F[*N][([A1[,A2]...])]	9.2.1
SUBROUTINE S[([A1[,A2]...])]	9.3.1
BLOCK DATA [S]	9.4
PROGRAM S	9.5

ПРИЛОЖЕНИЕ 3

ОСНОВНЫЕ ВНЕШНИЕ ФУНКЦИИ

В данном приложении приведены основные внешние (библиотечные) функции языка ФОРТРАН (см. таблицу).

Таблица

Функция	Определение	Число параметров	Символическое имя	Тип параметров	Тип функции
1	2	3	4	5	6
Абсолютное значение 1)	|A|	1	ABS	вещ.	вещ.
			IABS	цел.	цел.
			DABS	дв.	дв.
			CABS	компл.	вещ.
Усечение	SIGN(A)*|A|	1	AINT	вещ.	вещ.
			INT	вещ.	цел.
			IDINT	дв.	цел.
Взятие остатка 2)	A1*(MOD A2)	2	AMOD	вещ.	вещ.
			MOD	цел.	цел.

1	2	3	4	5	6
			DMOD	дв.	дв.
Выбор наибольшего значения	MAX (A1, A2, ...)	>=2	AMAX0	цел.	вещ.
			AMAX1	вещ.	вещ.
			MAX0	цел.	цел.
			MAX1	вещ.	цел.
			DMAX1	дв.	дв.
Выбор наименьшего значения	MIN (A1, A2, ...)	>=2	AMIN0	цел.	вещ.
			AMIN1	вещ.	вещ.
			MIN0	цел.	цел.
			MIN1	вещ.	цел.
			DMIN1	дв.	дв.
Преобразование в плавающую форму	Преобразование от целого к вещественному	1	FLOAT	цел.	вещ.
Преобразование в фиксированную форму	Преобразование от вещественного к целому	1	IFIX	вещ.	цел.
Передача знака	SIGN(A2)*!A1!	2	SIGN	вещ.	вещ.
			ISIGN	цел.	цел.

1	2	3	4	5	6
			DSIGN	дв.	дв.
Положительная разность	A1—MIN(A1, A2)	2	DIM	вещ.	вещ.
			IDIM	цел.	цел.
Получение максимальной значащей части параметра двойной точности		1	SNGL	дв.	вещ.
Получение действительной части комплексного параметра		1	REAL	компл.	вещ.
Получение мнимой части комплексного параметра		1	AIMAG	компл.	вещ.
Преобразование вещественного параметра в форму двойной точности		1	DBLE	вещ.	дв.
Преобразование двух вещественных параметров в комплексную форму	A1+A2*(—1)**(1/2)	2	CMPLX	вещ.	компл
Получение комплексной величины, сопряженной с параметром		1	CONJG	компл.	компл
Экспоненциальная функция	E**A	1	EXP	вещ.	вещ.

1	2	3	4	5	6
			DEXP	дв.	дв.
			CEXP	компл.	компл
Натуральный логарифм 3)	LN(A)	1	ALOG	вещ.	вещ.
			DLOG	дв.	дв.
			CLOG	компл.	компл
Десятичный 3) логарифм	LG(A)	1	ALOG10	вещ.	вещ.
			DLOG10	дв.	дв.
Тригонометри- 4) ческий синус	SIN(A)	1	SIN	вещ.	вещ.
			DSIN	дв.	дв.
			CSIN	компл.	компл
Тригонометри- 4) ческий косинус	COS(A)	1	COS	вещ.	вещ.
			DCOS	дв.	дв.
			CCOS	компл.	компл
Гиперболический тангенс	TANH(A)	1	TANH	вещ.	вещ.
квадратный 5) корень	A**(1/2)	1	SQRT	вещ.	вещ.
			DSQRT	дв.	дв.

1	2	3	4	5	6
			CSQRT	компл.	компл
6) Арктангенс	ARCTAN(A)	1	ATAN	вещ.	вещ.
			DATAN	дв.	дв.
	ARCTAN(A1/A2)	2	ATAN2	вещ.	вещ.
			DATAN2	дв.	дв.
Генератор случайного числа	Вычисление случайного числа из равномерного распределения от 0 до 1	2	RAN	цел.	вещ.

[1] Абсолютным значением комплексного числа (X, Y) является действительное значение (X**2+4**2)**(1/2).

[2] Функции AMOD(A1, A2), MOD(A1, A2), DMOD(A1, A2) определяются как A1 — [A1/A2]*A2, где [A1/A2] — наибольшее целое, величина которого не превосходит величины A1/A2 и имеющее тот же знак, что и A1/A2.

[3] Аргумент функции должен быть больше нуля.

[4] Аргумент функции должен быть в радианах.

[5] Аргумент функции должен быть больше или равен нулю.

[6] Результат функции получается в радианах.

Сокращения:

дв. — двоичный;
цел. — целый;
компл. — комплексный;
вещ. — вещественный.

ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Операционная система ФОДОС-2
   Программа пакетной обработки
   Руководство оператора
2. ФОРТРАН/ФОДОС-2

Транслятор с ФОРТРАНа
Руководство программиста
3. ФОРТРАН/ФОДОС-2
Библиотека ФОРТРАНа
Руководство программиста
4. ФОРТРАН/ФОДОС-2
Диалоговый отладчик
Руководство программиста

ТРАНСЛЯТОР С ФОРТРАНА

РУКОВОДСТВО ПРОГРАММИСТА

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ

1.1. Назначение

Транслятор с языка ФОРТРАН (файл FORTRA.SAV) предназначен для перевода программы, написанной на языке ФОРТРАН [5], в эквивалентную ей объектную программу, пригодную для дальнейшей обработки ее программами операционной системы ФОДОС-2.

Транслятор с языка ФОРТРАН работает под управлением любого монитора (SJ, FB или XM) дисковой операционной системы ФОДОС-2 и выполняет следующие функции:

— управление процессом трансляции с помощью переключателей командной строки;
— синтаксический и семантический контроль и выдачу сообщений об ошибках;
— получение объектного модуля, удовлетворяющего требованиям операционной системы ФОДОС-2;
— получение листинга программы с информацией о распределении памяти;
— определение символических имен библиотеки ФОРТРАНа, которые используются в программе пользователя.

Создаваемые транслятором объектные модули удовлетворяют требованиям программы «Редактор связей» [1], что дает возможность формирования программ с оверлейной структурой.

1.2. Условия применения

1.2.1. Требования к техническим средствам

Для работы транслятора с языка ФОРТРАН необходима следующая минимальная конфигурация технических средств:

— центральный процессор;
— оперативное запоминающее устройство емкостью 16К слов для работы под управлением мониторов SJ и FB и 32К

слов и диспетчер памяти для работы под управлением монитора XM;
— устройство ввода-вывода информации на гибких магнитных дисках или на жестком магнитном диске;
— таймер;
— терминал.

Транслятор с ФОРТРАНа обеспечивает также обслуживание следующих устройств:
— накопителя на магнитной ленте типа СМ 5300.01;
— построчно-печатающего устройства.

1.2.2. Требования к программным средствам

Для работы транслятора с языка ФОРТРАН необходимы следующие компоненты операционной системы ФОДОС-2:
FMON<YY>.SYS — монитор системы;
SWAP.SYS — файл свопинга;
TT.SYS — драйвер терминала;
<XX>.SYS — драйвер диска;
K13.SAV — редактор текста;
LINK.SAV — редактор связей
где YY — SJ, FB или XM;
XX — DX, DY, DW, MX, MY, или DXX, DYX, MXX, MYX, DWX.

Для трансляции необходимо, чтобы исходная программа находилась на устройстве файловой структуры.

2. ОБРАЩЕНИЕ К ПРОГРАММЕ

Транслятор с языка ФОРТРАН работает под управлением дисковой операционной системы реального времени ФОДОС-2 [1].

2.1. Вызов транслятора по команде RUN

Для вызова транслятора следует загрузить операционную систему ФОДОС-2 [3] и подать с терминала следующую команду:

.RUN уст:FORTRA<ВК>

где уст — устройство, на котором находится файл FORTRA.SAV.

Для вызова транслятора с системного устройства (носитель, на котором находится система ФОДОС-2) достаточно подать команду .R FORTRA <ВК>

После вызова транслятор выводит звездочку и ожидает ввода командной строки. Если после этого нажать клавишу

<ВК>, то транслятор выводит номер своей версии. Подчеркнутый текст печатается ЭВМ.

Формат командной строки:
обспф, листспф=спф [,... спф] [/прк] <ВК>
где обспф — спецификация объектного файла (устойство, имя и тип файла);
листспф — спецификация файла листинга;
спф — спецификация входного файла;
/прк — переключатель (табл. 3).

В командной строке можно указать до шести входных файлов. Входной файл — это файл, содержащий программный модуль на языке ФОРТРАН.

Значения элементов спецификации файла по умолчанию приведены в табл. 1.

Таблица 1

ЗНАЧЕНИЯ ЭЛЕМЕНТОВ СПЕЦИФИКАЦИИ ФАЙЛА ПО УМОЛЧАНИЮ

Файл	Устройство	Имя файла	Тип
1	2	3	4
Объектный	DK:	Необходимо указать:	.OBJ
Листинг	Устройство, назначенное для объектного модуля	То же	.LST
Входной	Для первого файла DK: для последующих то же, что для предыдущего	—//—	.FOR

Обозначения периферийных устройств приведены в табл. 2. Для получения объектного файла устройства TT: и LP: не используются.

Таблица 2

ОБОЗНАЧЕНИЯ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ

Имя устройства	Устройство
1	2
DK:	Устройство по умолчанию для всех файлов; первоначально то же, что и SY:
DKN:	DK: с указанием номера привода, N от 0 до 7
DPN:	Пакет магнитных дисков (29 Мбайт), N от 0 до 7
MYN:	Гибкие диски, N от 0 до 3 (удвоенная плотность записи)

1	2
MXN:	Гибкие диски, N от 0 до 3 (одинарная плотность записи)
LP:	Построчно-печатающее устройство
MTN:	Магнитная лента, N от 0 до 7
PC:	Перфоленточное устройство ввода-вывода
SY:	Системное устройство
SYN:	SY: с указанием номера привода N, N от 0 до 7
TT:	Терминал

После выполнения трансляции печатается звездочка и ожидается ввод новой командной строки.

Для передачи управления монитору следует подать команду СУ/С после окончания трансляции программы или дважды СУ/С — во время трансляции.

Для повторного пуска транслятора следует подать команду монитора REENTER <ВК>.

2.2. Вызов транслятора по команде FORTRAN

Вызвать транслятор с системного устройства можно по команде монитора FORTRAN.

Команда FORTRAN задается в виде полной строки

.FORTRAN [/прк] спф [,...,спф] [/прк] <ВК>

или краткой строки

.FORTRAN [/прк] <ВК>

FILES? спф [,... спф] [/прк] <ВК>

где /прк — переключатель (табл. 3);

спф — спецификация входного файла.

Если в командной строке задается несколько входных файлов, то их разделяют знаками плюс. Создается выходной файл типа .OBJ с именем первого входного файла. Если входные файлы разделить запятыми, то транслятор создаст объектный файл для каждого файла.

3. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

Входные и выходные данные

Входными данными для работы транслятора с языка ФОРТРАН являются программы на языке ФОРТРАН. Программа может состоять из одного или нескольких программных модулей. Каждый модуль может транслироваться независимо. Результатом трансляции являются два выходных

файла: объектный файл и файл листинга. Объектные модули обрабатываются программой редактор связей для получения загрузочного модуля.

3.1. Программный модуль

Программный модуль представляет собой последовательность строк на языке ФОРТРАН. Обычная строка состоит из 72 символов, расширенная — из 80.

Пример записи программного модуля приведен на рис. 1. Запись программного модуля разбита по полям. Каждое поле помечено цифрой.

<1> — поле метки предложения (позиции 1—5)

Буква С в позиции 1 указывает, что данная строка является комментарием.

<2> — поле признака продолжения (позиция 6)

<3> — поле предложения (позиции 7—72)

<4> — поле идентификации (позиции 73—80)

Правила записи программ на языке ФОРТРАН приведены в документе [5].

<1> 1 5	<2> 6	<3> 7 72	<4> 73 80
C		DEMONSTRATION TEST	
C			
		TYPE 100	
100		FORMAT ('***** FORTRAN DEMONSTRATION	
	+	TEST ******')	
		A = 3.14159	
		A = A/3.14159	
		IF (A.NE.1.0) TYPE 200	
200		FORMAT ('0ERROR IN FLOATING POINT	
	+	ARITHMETIC TEST.')	
		I = A	
		I = I*8/4	
		IF (I.NE.2) TYPE 300	
300		FORMAT ('0ERROR IN INTEGER ARITHMETIC	
		TEST.')	
		TYPE 400	
400		FORMAT ('0INSTALLATION SUCCESSFUL IF NO	
	+	ERROR MESSAGES' ,/' WERE PRINTED ABOVE.')	
		TYPE 500	
500		FORMAT ('0***** FORTRAN DEMONSTRATI-	
	+	ON TEST COMPLETE *****')	
		CALL EXIT	
		END	

Рис. 1

3.2. Объектный модуль, полученный после трансляции, удовлетворяет требованиям операционной системы ФОДОС-2 и пригоден для дальнейшей обработки программами операционной системы ФОДОС-2.

Структура объектного модуля описана в документе [1].

3.3. Листинг трансляции содержит следующие поля:

— поле режима трансляции;
— поле исходной программы;
— поле диагностики;
— поле карты памяти;
— поле генерированного кода;
— поле статистики.

Переключатели командной строки задают получение любой комбинации этих полей листинга.

Поле генерированного кода может быть получено в одном из двух кодов — поточном или нанизанном.

В примере 1 приведен полный листинг программы пользователя. Каждое поле листинга помечено цифрой. Поле генерированного кода содержит поточный код.

В примере 2 приведено поле генерированного кода программы примера 1. Поле генерированного кода содержит нанизанный код.

Пример 1:

```
<1>  FORTRAN        B02.00     WED 10 JUN 84 09:34:55
,TT:=DK0:PR5.FOR/O/A/L:7
OPTIONS IN EFFECT:
                      SOURCE
                      MAP
                      CODE-
                    NOLEAPYEAR
                    NOREADONLY
                      LRECL=0136
                      STAT
                      ISNS
                    NOCOL80
                      USRSWAP
                      DIAGNOSE
                    NOINTEGER*4
                      NLCHN=06
                    NODEBUG
                      VECTOR
                    NOWARN
                      CODE:EIS
                      LOG
```



```
<2>
    FORTRAN     B02.00     WED 10 JUN 84 09:34:55      PAGE 001
    0001        COMPLEX IMAG
    0002        DATA INT/100/
    0003        REAL=INT/2 + 9.
    0004        IMAG=CMPLX(REAL,3. 21)
    0005        WRITE(5,10) IMAG
    0006        FORMAT (1X,2F8.5)
    0007        END
    .MAIN.
<3>
FORTRAN          DIAGNOSTICS FOR PROGRAM UNIT .MAIN.
IN LINE 0005, ERROR: REFERENCE TO UNDEFINED STATEMENT LABEL
IN LINE 0006, ERROR: UNLABELED FORMAT STATEMENT
<4>

FORTRAN          STORAGE MAP FOR PROGRAM UNIT .MAIN.
LOCAL VARIABLES, .PSECT $DATA, SIZE=000020( 8. WORDS)
NAME TYPE OFFSET    NAME  TYPE OFFSET    NAME  TYPE  OFFSET
IMAG C*8  000004    INT   I*2  000002    REAL  R*4   000014
SUBROUTINES,FUNCTIONS,STATEMENT AND PROCESSOR DEFINED FUNCTIONS:
NAME TYPE    NAME TYPE    NAME TYPE    NAME TYPE    NAME TYPE
CMPLX C*8

<5>
FORTRAN          GENERATED CODE FOR PROGRAM UNIT .MAIN.
                                          ;STATEMENT #0003
000006     MOV     #3,@$AOTS                      R 012345
000014     MOV     INT,R1                         R   2345
000020     SXT     R0                             R   2345
000022     DIV     #2,R0                          R   2345
000026     MOV     R0,(SP)                        R   2345
000030     JSR     PC,$CVTIF                      R 012345
000034     CLR     (SP)                           R 012345
000036     MOV     #40640, (SP)                   R 012345
000042     JSR     PC,$ADDF                       R 012345
000046     MOV     (SP)+,REAL                     R 012345
000052     MOV     (SP)+,REAL+2                   R 012345
                                          ;STATEMENT #0004
000056     INC     @$AOTS                         R 012345
000062     MOV     #C$CJ,R5                       R 012345
000066     JSR     PC,CMPLX                       R 012345
           :
           :
           :
                                          ;STATEMENT #0007
000126     INC     @$AOTS                         R 012345
000132     RTS     PC                             R 012345
<6>
COMPILATION STATISTICS:
SYMBOL   TABLE   SIZE:    00078 WORDS
INTERNAL FORM    SIZE:    00082 WORDS
TREE     DYNAMIC MEMORY: 19625 WORDS
COMPILATION TIME: 00:00:43
```

<1> Поле режима трансляции содержит информационную строку, командную строку и список условий, определяющих режим трансляции. Информационная строка определяет:

— наименование транслятора и его версию;
— день недели и дату (число, месяц, год);
— время дня (час, минута, секунда).

Условие в списке режима трансляции, начинающееся с NO, не выполняется.

<2> Поле исходной программы содержит информационную строку с номером страницы листинга, текст тела программного модуля и заголовок программного модуля. Текст тела программного модуля разбивается по страницам (стандартная длина страницы — 56 строк) и содержит номера внутренней последовательности. Сообщения об ошибках вида ****Y выводятся непосредственно после строки, в которой обнаружена ошибка (п. 4.1).

<3> Поле диагностики формируется в случае обнаружения ошибок. Первая строка содержит наименование транслятора, наименование данного поля и заголовок программного модуля. Далее следуют сообщения об ошибках. Перечень возможных сообщений об ошибках приведен в р. 4.

<4> Поле карты памяти

Первая строка поля карты памяти содержит наименование транслятора, наименование поля и заголовок программного модуля. Далее следует информация о локальных символических именах, используемых в программном модуле, блоках COMMON, массивах, подпрограммах и функциях.

<5> Поле генерированного кода

Первая строка поля генерированного кода содержит наименование транслятора, наименование поля, заголовок программного модуля. Далее следует генерированный поточный код. Поточный код — это объектный код программы на языке АССЕМБЛЕР [1], т. е. транслятор с ФОРТРАНа переводит программу с языка ФОРТРАН на язык АССМБЛЕР и выдает объектный код трансляции на языке АССЕМБЛЕР (пример 1).

Поле генерированного кода может быть в одном из двух видов: поточном или нанизанном. Нанизанный код — это код, генерируемый в виде обращения к подпрограммам библиотеки ФОРТРАНа (пример 2). Левый столбец поля генерированного кода содержит смещение от базового адреса. Следующие два столбца содержат обращения к библиотеке

ФОРТРАНа (символические имена подпрограмм и параметры).

Для поточного кода при задании переключателя /DIAGNOSE или /B в командной строке, указываются регистры общего назначения, доступные для использования (правый столбец).

Код, генерированный для каждого оператора, помечается тем же номером внутренней последовательности, что и в поле исходной программы.

Генерированный код листинга содержит первое предложение, с относительным адресом 000006.

По первым двум относительным адресам (000002, 000004) транслятор формирует следующую последовательность операторов:

Для головного модуля

```
JSR     R4, $$OTI       (поточный код)
.WORD   NAMPTR
или
JSR     R4,$OTI         (нанизанный код)
.WORD   NAMPTR
```

Для подпрограммы

```
JSR     R4,$OTIS
.WORD   NAMPTR
```

В ячейке с именем NAMPTR содержится адрес двухсловного сегмента, в котором находится имя головного модуля или имя подпрограммы в коде RADIX-50. Для головного модуля по умолчанию используется имя .MAIN..

<6> Поле статистики

Первая строка поля статистики содержит наименование поля. Далее следует сообщение об использовании памяти во время трансляции и времени, затраченном на трансляцию.

Пример 2:

```
FORTRAN          GENERATED CODE FOR PROGRAM UNIT .MAIN.
STATEMENT #0003
000006   LSN$                #000003
000012   MOI$MS              $DATA+#000002
000016   DII$IS              #000002
000022   CFI$
000024   ADF$IS              #040640
000030   MOF$SM              $DATA+#000014
STATEMENT #0004
000034   ISN$
000036   REL$                $DATAP+#000010
000042   REL$                $DATA+#000014
000046   CAL$                #000002 CMPLX+#000000
000054   MOD$RM              $DATA+#000004
```

```
STATEMENT #0005
000060        ISN$
000062        FUD$
STATEMENT #0006
000064        ISN$
000066        FUD$
STATEMENT #0007
000070        ISN$
000072        RET$
```

3.4. Режим трансляции задается переключателями командной строки (табл. 3).

Таблица 3

ПЕРЕКЛЮЧАТЕЛИ ТРАНСЛЯТОРА

Переключатель		Назначение
.FORTRAN	.RFORTRA	
1	2	3
/ALLOCATE:M[1)]	—	Резервирует M блоков памяти на томе для объектного файла и файла листинга. Используется с переключателями /OBJECT и /LIST
/CODE:XXX[2)]	/I:XXX[2)]	Задает вид объектного кода
/DIAGNOSE	/B	Разрешает в листинге вывод расширения поля для поточного кода
/EXTEND	/E	Разрешает ввод поля идентификации исходной программы
/HEADER	/O	Разрешает в листинге вывод поля режима трансляции
/I4	/T	Резервирует два слова памяти для целых переменных, не имеющих явного указания длины. Значение целой переменной размещается в слове с младшим адресом. По умолчанию резервируется одно слово памяти
/LINENUMBERS	—	Разрешает вывод номеров внутренней последовательности в объектный код
/LIST[:спф]	—	Вывод листинга
/NOLINENUMBERS	/S	Запрещает вывод номеров внутренней последовательности в объектный код
/NOOBJECT	—	Запрещает вывод объектного файла
/NOSWAP	/U	Запрещает свопинг программы USR во время выполнения программы пользователя

1	2	3
/NOVECTORS	/V	Запрещает векторизацию многомерных массивов
/OBJECT[:спф]	—	Вывод объектного файла
/ONDEBUG	/D	Разрешает трансляцию строк отладки. По умолчанию строки отладки рассматриваются как комментарии
/RECORD:M	/R:M	Определяет максимальный размер записи в байтах для операторов форматного ввода-вывода (4<M<4095). По умолчанию M=136
/SHOW[:M][1)]	/L[:M][1)]	Задает вывод основных полей листинга (0<=M<=7)
/STATISTICS	/A	Разрешает в листинге вывод поля статистики
/SWAP	—	Разрешает свопинг программы USR во время выполнения программы пользователя
/UNITS:M[1)]	/N:M[1)]	Определяет максимальное число логических устройств (1<=M<=15), которые могут быть открыты одновременно. По умолчанию M=6
/VECTORS	—	Разрешает векторизацию многомерных массивов
/WARNIGS	/W	Разрешает вывод сообщений, предупреждающих об ошибках. Используется с переключателем /SHOW или /L
—	/H	Вывод вспомогательного файла FORTRA.HLP (список переключателей транслятора)
—	/Q	Запрещает в листинге вывод заголовка программного модуля (головного или FUNCTION, SUBROTINE и BLOCK DATA)
—	/X	Зарезервирован для расширения возможностей транслятора
—	/Z	Присваивает программным секциям, содержащим генерированный код: ($CODE или данные ($DATAP), значение признака доступа R0

1)
M — целое восьмеричное или десятичное число. Указателем десятичного числа является точка, следующая за числом.
2)
XXX — EIS, FIS или THR.

3.4.1. Управление полями листинга

Вывод основных полей листинга задается с помощью переключателя /SHOW[:M] или /L[:M].

Параметр M может принимать следующие значения:

0 или пусто — вывод поля диагностики;
1 или SRC — вывод поля исходной программы и поля диагностики;
2 или MAP — вывод поля карты памяти и поля диагностики;
4 или COD — вывод поля генерированного кода и поля диагностики.

Можно задать любую комбинацию указанных параметров, суммируя их числовые значения.

Например:

7 или ALL — вывод всех основных полей листинга.

По умолчанию (если переключатель не задан) осуществляется вывод следующих полей:
— исходной программы;
— диагностики;
— карты памяти.

Это эквивалентно заданию переключателя /SHOW:3 или /L:3.

Описание полей листинга приведено в п. 3.3.

3.4.2. Виды объектного кода

Вид генерированного объектного кода задается с помощью переключателя /CODE:XXX или /I:XXX.

Параметр XXX может принимать следующие значения:
EIS — поточный код с использованием команд расширенной арифметики (MUL, DIV, ASH, ASHC);
FIS — поточный код с использованием команд расширенной арифметики и плавающей запятой (MUL, DIV, ASH, ASHC, FADD, FSUB, FMUL, FDIV);
THR — нанизанный код. Этот код содержит основной базовый набор команд процессора.

Если программа пользователя не содержит арифметические операции с данными вида REAL*4, REAL*8 или COMPLEX*8, то:
1) поточный код всегда выполняется быстрее нанизанного;
2) различие по памяти между программами поточного и нанизанного кода незначительно.

Если программа содержит большое количество арифметических операций с данными вида REAL*4, REAL*8 и COMPLEX*8, то:
1) нанизанный код требует памяти намного меньше поточного;

2) скорость выполнения для обоих кодов примерно одинакова.

Сравнение поточного и нанизанного кодов приведено в табл. 4.

Таблица 4

СРАВНЕНИЕ ПОТОЧНОГО И НАНИЗАННОГО КОДОВ ДЛЯ ВЫРАЖЕНИЯ I=J*K+REAL

Нанизанный код	Поточный код	
1	2	
THR	FIS	EIS
1	2	3

MOI$MS J
MUI$MS K
	MOV J, R1
	MUL K, R1
	MOV R1,—(SP)
		MOV J, R1
		MUL K, R1
		MOV R1,—(SP)

CFI$	JSR PC, $CVTIF	JSR PC, $CVTIF

ADF$MS REAL [1]	MOV REAL+2,—(SP)
	MOV REAL,—(SP)
	FADD SP
		MOV RAEL+2,—(SP)
		MOV REAL,—(SP)
		JSR PC, $ADDF

CIF$ MOI$SM I	JSR PC, $CVTFI MOV (SP)+, I	JSPC, $CVTFI MOV (SP)+, I

1)

Для записи числа в форме с плавающей точкой нанизанный код требует два слова памяти плюс размер программы ADF$MS, в то время как поточный код FIS — пять слов, EIS — четыре. Это обеспечивает экономию памяти при использовании нанизанного кода для операций с плавающей точкой.

Транслятор с ФОРТРАНа размещает объектный код по трем именованным программным секциям .PSECT (табл. 5).

Таблица 5

РАЗМЕЩЕНИЕ ОБЪЕКТНОГО КОДА ПО СЕКЦИЯМ

Имя секции	Признаки	Содержимое
1	2	3
$CODE	RW, I, LCL, REL, CON	Генерированный код (нанизанный или поточный)

1	2	3
$DATAP	RW, D, LCL, REL, CON	Данные, которые не изменяются во время выполнения программы (константы, текстовые данные, спецификации формата, векторы массивов)
$DATA	RW, D, LCL, REL, CON	Данные, которые требуют изменения (переменные, область временной памяти и массивы, используемые в программе)

ПРИМЕЧАНИЕ. Для секций $CODE и $DATAP признак R0/RW задается переключателем /Z.

Общие объекты, описанные с помощью объявления COMMON, помещаются в именованные программные секции с признаками RW, D, GBL, REL, OVR.

Например, объявлению COMMON /X/ A, B, C соответствует следующая программная секция на языке АССЕМБЛЕР:

```
        .PSECT  X, RW, D, GBL, REL, OVR
A:      .BLKW   2
B:      .BLKW   2
C:      .BLKW   2
```

Объявлению COMMON C,B/X/A соответствуют следующие программные секции:

```
        .PSECT  $$$$., RW, D, GBL, REL, OVR
C:      .BLKW   2
B:      .BLKW   2
        .PSECT  X, RW, D, GBL, REL, OVR
A:      .BLKW   2
```

Непомеченному общему блоку COMMON присваивается имя .$$ $$..

Примеры листингов с различными видами объектных кодов приведены в п. 3.3.

3.4.3. Векторизация массивов

Управление способом размещения элементов многомерных массивов в памяти ЭВМ и управление способом поиска осуществляется с помощью переключателей /VECTORS, /NOVECTORS и /V. По переключателю /VECTORS упорядочивание элементов осуществляется путем векторизации массивов, по переключателю /NOVECTORS и /V — с использованием стандартной функции линеаризации.

Векторизация массива снижает время, необходимое для упорядочивания и обращения к элементам многомерного массива, за счет использования дополнительной памяти для хранения вектора массива. Транслятор векторизует массив на основании соотношения объема памяти, требуемого вектору массива, и объема памяти, требуемого для хранения массива. Если это соотношение превышает 25%, вектор массиву не придается, и в этом случае используется стандартная функция линеаризации. Виртуальные массивы с регулируемыми размерами не векторизуются. Векторизованные массивы помечаются в поле карты памяти (VEC).

Пользователь должен следить за тем, чтобы обращения к элементам векторизованного массива не выходили за пределы этого массива во избежание непредвиденных результатов при выполнении программы (например, прерывания). Особое внимание следует уделять массивам, передаваемым в подпрограммы, для которых список границ, указанный в подпрограмме, отличается от списка границ, указанного в программе. В этом случае создаются два набора векторов: для программы и подпрограммы. В подпрограмме векторизуется та часть массива, которая объявлена в данной подпрограмме.

Если массив векторизован, то элемент массива можно определить, используя указатели (элементы) вектора, приписанного к данному массиву. Например, векторизованному двумерному массиву B(5, 6) приписывается вектор P. Процесс придания вектора P массиву показан в табл. 6. Положение элемента B(M, N) относительно начала массива вычисляется как P(N)+M.

Таблица 6

ПРИДАНИЕ ВЕКТОРА МАССИВУ

Массив		Соответствующий вектор	
1		2	
B(1,1)	P(1)	0	P(1)
B(2,1)		5	P(2)
B(3,1)		10	P(3)
B(4,1)		15	P(4)
B(5,1)		20	P(5)
B(1,2)	P(2)	25	P(6)
B(2,2)			
B(3,2)			
⋮			
⋮			
⋮			

1		2
B(1,6) B(2,6) : : : B(5,6)	P(6)	

Например, положение элемента B(4, 3) определяется как P(3)+4=10+4=14

Объем памяти, требуемый для вектора массива, может быть вычислен как сумма верхних границ по измерениям массива за исключением первого.

Для уменьшения памяти, резервируемой под вектор, следует выполнять следующие условия:

1) в описании массива верхнюю границу по измерению, имеющую максимальное значение, следует помещать первой в списке границ, так как по первому измерению векторизация не осуществляется. Например, INTEGER A(350, 10) требует для вектора 10 слов, INTEGER A(10, 350) — 350 слов;

2) при задании массивов одного типа необходимо соблюдать одинаковый порядок следования верхних границ по измерениям, что позволяет массивам использовать один и тот же вектор. Например, DIMENSION A(10, 10), B(10, 20). В этом случае массивы A и B делят вектор из 20 слов: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, массив A использует только первые десять из них;

3) векторизацию можно запретить с помощью переключателя /NOVECTORS или /V. В этом случае вектор массиву не приписывается, программа пользователя выполняется медленнее, чем при векторизации.

3.4.4. Номера внутренней последовательности

Генерация номеров внутренней последовательности задается с помощью переключателей /LINENUMBERS, /NOLINENUMBERS и /S. Код, генерированный для каждого оператора, помечается тем же номером внутренней последовательности, что и в поле исходной программы листинга. Эти номера указываются во время трассировки ошибок [4].

Использование переключателя /NOLINENUMBERS позволяет сократить объем памяти для генерированного кода и снизить время выполнения программы пользователя. Однако, во время трассировки ошибок будет отсутствовать информа-

ция о номере строки, в которой произошла ошибка во время выполнения программы.

3.5. Логические и физические устройства

Обращение к устройствам ввода-вывода осуществляется через их логические номера. Номера устройств и поставленные им в соответствие физические устройства, имена и типы файлов определяются транслятором автоматически (табл. 7). Такое соответствие может быть изменено по команде монитора ASSIGN до выполнения программы пользователя или подпрограммой ASSIGN библиотеки ФОРТРАНа или IASIGN библиотеки системных подпрограмм во время выполнения программы. Логические номера, отличные от перечисленных в табл. 7, присваиваются любым устройствам ввода-вывода по усмотрению пользователя. Логическим устройствам с номерами от 10 до 99 автоматически ставятся в соответствие имена файлов FTN10.DAT, . . . , FTN99.DAT.

Программа пользователя может обращаться к логическим устройствам в любом порядке, однако, число логических устройств, одновременно задействованных, не может превышать шести. Это ограничение можно снять, если в командную строку включить переключатель /UNITS:M или /N:M.

Запись оператора форматного ввода READ F, K эквивалентна записи READ (1, F)K.

Таким образом, номер логического устройства «1» можно опускать. Другие номера логических устройств опускать не разрешается. В операторах ACCEPT, TYPE, PRINT номер логического устройства не указывается, так как он определен системно и равен пяти, семи и шести соответственно.

Таблица 7

НОМЕРА ЛОГИЧЕСКИХ УСТРОЙСТВ

Номер устройства	Физическое устройство	Имя файла
1	2	3
1	Системное устройство (SY:)	FTN1.DAT
2	Устройство по умолчанию (резерв)	FTN2.DAT
3	Устройство по умолчанию (резерв)	FTN3.DAT
4	Устройство по умолчанию (резерв)	FTN4.DAT
5	Клавиатура терминала (TT:)	FTN5.DAT
6	Построчно-печатающее устройство (LP:)	FTN6.DAT
7	Печать терминала (TT:)	FTN7.DAT
8	Перфоленточное устройство ввода (PC:)	FTN8.DAT
9	Перфоленточное устройство вывода (PC:)	FTN9.DAT

3.6. Связь программ и подпрограмм

Связь между программой и подпрограммой на языке ФОРТРАН осуществляется аналогично связи на языке АССЕМБЛЕР. Управление подпрограмме передается по команде JSR PC, SUBR

где SUBR — глобальное имя, точка входа в подпрограмму.

Регистр R5 содержит начальный адрес списка параметров. Формат списка параметров приведен в табл. 8. В качестве адреса отсутствующего параметра в список параметров заносится значение «—1». Отсутствующий параметр отмечается в операторе CALL последовательными запятыми. Например, CALL SUB(A,,B).

Таблица 8

Не определено	Число параметров
1	2
Адрес 1-го параметра	
Адрес 2-го параметра	
Адрес N-го параметра	

Выход из подпрограммы и передача управления в вызывающую программу осуществляется по команде RTS PC.

Пример использования подпрограммы на языке АССЕМБЛЕР с программой на языке ФОРТРАН.

Подпрограмма IADDR на языке АССЕМБЛЕР вычисляет сумму целых чисел, передаваемых из вызывающей программы, и возвращает результат в вызывающую программу. Обращение к подпрограмме IADDR имеет вид:

CALL IADDR (N1, N2, . . . , NN, I)

где N1, N2, . . . , NN — целые числа, сумму которых нужно вычислить;

I — переменная или элемент массива, принимающая значение суммы целых чисел.

Подпрограмма на языке АССЕМБЛЕР


```
        .GLOBL IADDR
IADDR:  MOV (R5)+, R0
        CLR R1
        DECB R0
1$:     ADD @(R5)+, R1
        DECB R0
        BNE 1$
        MOV R1, @(R5)+
        RTS PC
```

Последовательность обращений на языке ФОРТРАН

```
      EXTERNAL IADDR
      CALL IADDR(1, 5, 7, I)
      CALL IADDR(15, 30, 10, 20, 5, J)
```

вызовет присвоение переменной I значения 13, переменной J — 80.

Подпрограмма на языке АССЕМБЛЕР, вызванная из программы на языке ФОРТРАН, не обязана сохранять содержимое регистров общего назначения, но должна восстанавливать содержимое стека так, чтобы каждому занесению в стек соответствовала выборка из стека до выхода из подпрограммы.

Программы на языке АССЕМБЛЕР, которые вызывают подпрограммы на языке ФОРТРАН, должны запоминать содержимое нужных регистров до обращения к подпрограмме и восстанавливать их, если необходимо, после выхода из подпрограммы.

Основные внешние функции языка ФОРТРАН помещают результат в регистры общего назначения следующим образом: результат целого или логического типа помещается в R0; результат вещественного типа помещается в R0, R1 (старшие разряды — в R0, младшие разряды — в R1); результат двойной точности помещается в R0—R3 (самые младшие разряды — в R3); результат комплексного типа помещается в R0—R3 (старшие разряды действительной части — в R0, младшие разряды — в R1, старшие разряды мнимой части — в R2, младшие разряды — в R3).

3.7. Получение программ в абсолютном формате

Система ФОРТРАН предусматривает получение загрузочного модуля программы в абсолютном двоичном формате для выполнения его на ЭВМ с минимальным набором периферийных устройств без использования операционной системы ФОДОС-2. Такая ЭВМ должна иметь минимум 4К слов памяти, терминал и перфоленточное устройство ввода для загрузки

программы. В программе на языке ФОРТРАН допустим ввод-вывод только с терминала. Другие устройства ввода-вывода должны поддерживаться подпрограммами, написанными пользователем.

Для получения программы в абсолютном формате необходимо выполнить трансляцию исходной программы как обычно и связать полученный объектный модуль с модулем UNI.OBJ библиотеки ФОРТРАНа (глобальное имя модуля $SIMRT), указав в командной строке программы редактор связей переключатели /L, /I, /F.

Следующая последовательность команд позволяет получить загрузочный модуль в формате LDA, который выводится на перфоленту

```
.R LINK <BK>
*PC:=DY:LOAD.OBJ/L/I/F <BK>
LIBRARY SEARCH? $SIMRT <BK>
LIBRARY SEARCH? <BK>
```

ПРИМЕЧАНИЕ. Необходима осторожность при использовании библиотеки ФОРТРАНа, так как она должна соответствовать возможностям ЭВМ.

3.8. Основные внешние функции

Для основных внешних функций, перечисленных в табл. 9, поточный код генерируется без обращения к подпрограмме по имени функции.

Таблица 9

Внешняя функция	Число параметров	Символическое имя	Тип параметра	Тип функции
1	2	3	4	5
Абсолютное значение	1	IABS	цел.	цел.
Передача знака	2	ISIGN	цел.	цел.
Взятие остатка	2	MOD	цел.	цел.
Выбор наименьшего значения	2	MIN0	цел.	цел.
Выбор наибольшего значения	2	MAX0	цел.	цел.
Преобразование в фиксированную форму	1	IFIX	вещ.	цел.
Преобразование в плавающую форму	1	FLOAT	цел.	вещ.
Получение действительной части комплексного параметра	1	REAL	компл.	вещ.

1	2	3	4	5
Преобразование вещественного параметра в форму двойной точности	1	DBLE	вещ.	дв.
Получение максимальной значащей части параметра двойной точности	1	SNGL	дв.	вещ.

Для обращения к внешней функции как к подпрограмме необходимо объявить имя этой функции как внешнее.

Например, для оператора I=IABS(J) транслятор генерирует следующий поточный код

```
      MOV J, I
      BPL 1$
      NEG I
1$:
      .
      .
      .
```

Для операторов

```
      EXTERNAL IABS
      I=IABS(J)
```

генерируется код, эквивалентный следующему:

```
      .GLOBL IABS
      MOV #J, -(SP)
      MOV #I, -(SP)
      MOV SP, R5
      JSR PC, IABS
      CMP (SP)+, (SP)+
      MOV R0, I
```

3.9. Организация памяти во время выполнения программы

Транслятор с ФОРТРАНа использует для своей работы всю оперативную память, которую предоставляет ему система. На рис. 2 показана организация памяти для 8К слов во время выполнения программы пользователя для случаев свопируемой и резидентной USR.

Если программа обслуживания прерываний, написанная пользователем, связана с программой на языке ФОРТРАН и предполагается осуществлять свопинг USR, то программу следует располагать выше зоны свопинга USR. USR загружается выше векторов прерываний, начиная с 1000 адреса, и занимает 2К слов.

Некоторые сегменты памяти во время работы программы имеют постоянный размер. Они включают резидентный монитор, рабочую область библиотеки ФОРТРАНа, стек, векторы прерываний и USR, если она резидентна.

Другие сегменты изменяют свой размер. Каждая программа пользователя имеет свой размер. Драйверы устройств, таблица каналов и буферы ввода-вывода размещаются в памяти динамически. Резидентны только драйверы используемых устройств. Буферы ввода-вывода размещаются и удаляются по требованию (при выполнении операторов ввода-вывода).

Объем памяти, отводимый под перечисленные выше сегменты переменной длины, ограничен. Для системы с объемом памяти 8К слов под переменные сегменты отводится приблизительно 3К слов, если USR резидентна, и 5К слов — если USR свопируема.

Если для работы программы недостаточно имеющегося объема памяти, следует уменьшить длину одного из переменных сегментов. Уменьшение числа периферийных устройств, используемых программой пользователя, уменьшает число драйверов, резидентно находящихся в памяти. Если программа пользователя после окончания ввода-вывода закрывает файл по подпрограмме библиотеки ФОРТРАНа CLOSE, то буфер ввода-вывода освобождается и программа пользователя может занимать эту часть памяти.

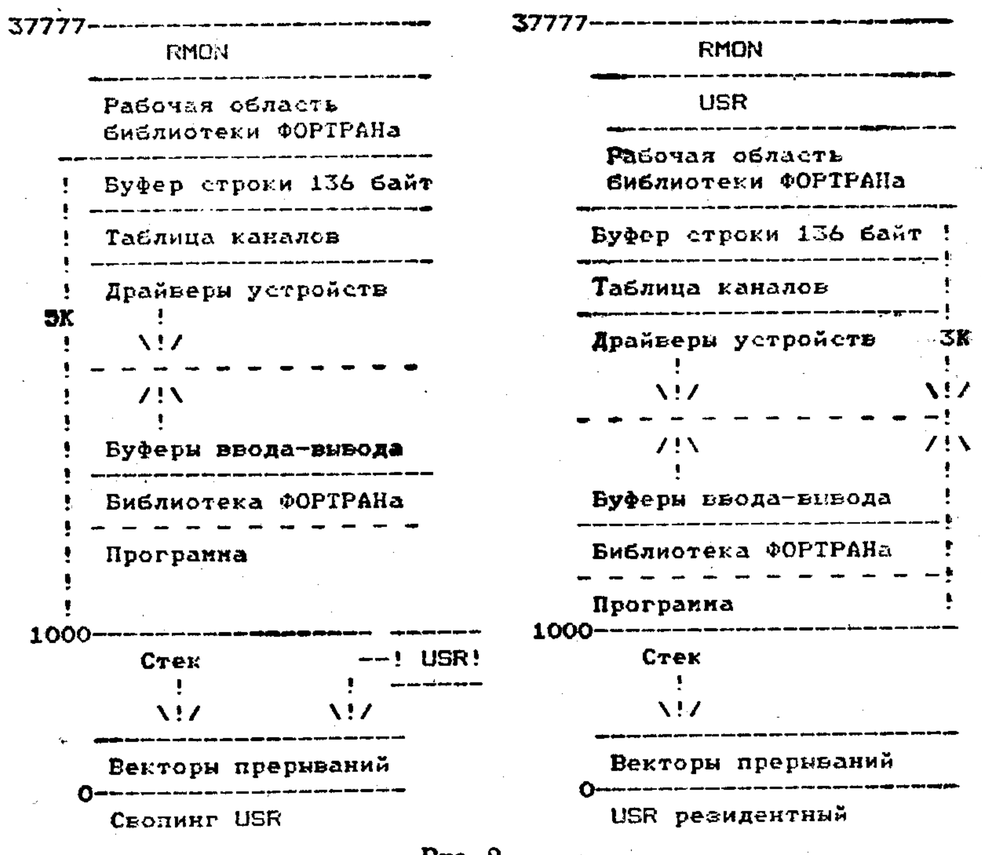


Рис. 2

3.9.1. Оверлеи

Создание программ с оверлейной структурой позволяет пользоваться практически неограниченной памятью.

Для создания программ с оверлейной структурой необходимо программу сегментировать, т. е. разбить на части (сегменты). Оверлейная структура состоит из корневого сегмента и одной или более оверлейных областей.

Корневой сегмент содержит головной модуль программы, неименованный общий блок и программную секцию .PSECT с признаком GBL, на которую есть ссылки более чем из одного сегмента. Корневой сегмент может содержать также несколько модулей-подпрограмм и/или модулей-функций.

Оверлейная область — это область памяти, где размещаются два или более оверлейных сегментов и только один из них может быть резидентным в текущий момент времени. Оверлейный сегмент содержит одну или более подпрограмм или функций.

Во время работы, если сделан вызов программы, содержащейся в оверлейном сегменте, осуществляется проверка, является ли данный сегмент резидентным в своей оверлейной области. Если сегмент не является резидентным, то он считывается из файла загрузочного модуля программы в оверлейную область, при этом разрушается предыдущий оверлейный сегмент в этой оверлейной области.

Размер оверлейной области равен размеру наибольшего оверлейного сегмента. Недопустимо из одного оверлейного сегмента вызывать подпрограмму, размещенную в другом оверлейном сегменте той же оверлейной области или в оверлейной области с меньшим номером, чем область вызывающей подпрограммы.

Головной модуль должен быть помещен в корневой сегмент.

В условиях оверлейной структуры вызов подпрограммы и функции должен относиться только к:

— библиотечной подпрограмме или функции ФОРТРАНа (например, ASSIGN, DCOS);
— подпрограмме на языке АССЕМБЛЕР или ФОРТРАН, находящейся в корневом сегменте;

— подпрограмме на языке АССЕМБЛЕР или ФОРТРАН, находящейся в том же оверлейном сегменте, что и вызывающая подпрограмма;
— подпрограмме на языке ФОРТРАН или АССЕМБЛЕР, находящейся в сегменте, номер оверлейной области которого больше, чем в вызывающей подпрограмме.

При оверлейной структуре блоки COMMON должны помещаться так, чтобы они были резидентными при обращении к находящимся в них переменным.

Неименованный блок COMMON всегда резидентен, поскольку он всегда размещается в корневом сегменте. Все именованные блоки COMMON должны размещаться или в корневом сегменте, или в сегменте с наименьшим номером области из всех сегментов, использующих этот блок COMMON. К именованному блоку COMMON не может быть обращения из двух сегментов одной и той же области, если блок COMMON не определен в сегменте с меньшим номером области.

Редактор связей автоматически помещает блок COMMON в корневой сегмент, если к нему обращается головная программа или подпрограмма, которая располагается в корневом сегменте. В противном случае редактор связей помещает блок COMMON в первый сегмент, описанный в командной строке редактора связей, в котором осуществляется определение этого блока COMMON.

Все блоки COMMON, значения переменных в которых задаются объявлениями DATA, должны описываться в тех же оверлейных сегментах, где они помещаются.

Более подробно о построении оверлейных структур описано в документе [2].

3.9.2. Оверлеи расширенной памяти

Можно использовать редактор связей для создания оверлейной структуры, использующей расширенную память, для привилегированных или виртуальных заданий. Для этого необходим монитор XM и диспетчер памяти.

Оверлейная структура расширенной памяти отличается от обычной оверлейной структуры тем, что оверлеи расширенной памяти могут находиться в расширенной памяти одновременно. Это отличие позволяет ускорить выполнение программы, так как оно уменьшает число обращений к вспомогательному тому памяти.

Ограничения на оверлеи расширенной памяти те же, что и на обычные оверлеи.

Следующая командная строка иллюстрирует использование оверлеев расширенной памяти вместо обычных оверлеев

для создания привилегированного задания:

```
.LINK/PROMPT/EXE:LOAD MAIN+SUBI <BK>
_
*SUB2/V:1/C  <BK>
_
*SUB3/V:1/C  <BK>
_
*SUB4/V:2/C  <BK>
_
*SUB5/C      <BK>
_
*SUB6/V:2/C  <BK>
_
*SUB7//  <BK>
_
```

3.10. Виртуальные массивы

Все виртуальные массивы, объявленные в программе, располагаются в неименованной программной секции .VSECT с признаком CON. Виртуальные массивы помещаются и обрабатываются в области расширенной памяти, доступ к которой осуществляется за минимально короткое время.

Преобразование программы для работы с виртуальными массивами осуществляется путем замены объявления DIMENSION на VIRTUAL.

Для этого необходимо:
— определить массивы, подлежащие преобразованию в виртуальные;
— определить объявления типа, в которых указаны массивы подлежащие преобразованию;
— заменить объявление DIMENSION, на VIRTUAL для массивов, подлежащих преобразованию, перенеся список границ из объявления типа в объявление VIRTUAL, если они не были указаны в DIMENSION;
— удалить из объявления типа переносимые в объявление VIRTUAL списки границ;
— оттранслировать программу;
— убедиться, что параметры виртуальных массивов правильно переданы подпрограммам:

1) если список параметров вызова подпрограмм включает неиндексированное имя виртуального массива, то список параметров операторов SUBROUTINE или FUNCTION должен иметь неиндексированное имя виртуального массива в позиционно соответствующем ему формальном параметре. Подпрограмме предоставляется доступ к виртуальному массиву.

Объявление виртуального массива в подпрограмме должно быть по размеру совместимо с его объявлением в вызывающей программе. Все изменения, которые появились в виртуальном массиве во время выполнения программы, сохраняются при возврате управления вызывающей программе. При передаче массивов как параметров необходимо, чтобы позиционно совпадающие параметры были переданы оба как виртуальные или как не виртуальные. Несовпадение типов массивов не обнаруживается ни во время трансляции, ни во время выполнения;

2) если список параметров обращения к подпрограмме включает обращение к элементу виртуального массива, то соответствующий формальный параметр в заголовке SUBROUTINE или FUNCTION должен быть невиртуальной переменной. Присвоение значения формальному параметру в подпрограмме не изменяет хранимого значения элемента виртуального массива в вызывающей программе. Для изменения значения этого элемента вызывающая программа должна содержать операторы присваивания, относящиеся непосредственно к данному элементу.

Следующий пример демонстрирует процедуру преобразования программы с невиртуальными массивами в программу с виртуальными массивами

```
      DIMENSION A(1000,20)
      INTEGER*2 V(1000)
      DATA V/1000*0/
      CALL ABC (A, V, 1000, 20)
      WRITE (2,*) (A(I,1),I=1,1000)
      END
      SUBROUTINE  ABC (X, Y, N, M)
      DIMENSION   X(N, M)
      INTEGER*2   Y(N)
      DO    10,I=1,N
10    X(I,1)=Y(I)
      RETURN
      END
```

В этой программе два массива A и V. Массив A указан в объявлении DIMENSION, тип данных задан по умолчанию. Для преобразования массива A в виртуальный достаточно заменить DIMENSION на VIRTUAL. Массив V указан в объявлении типа. Для преобразования его в виртуальный массив нужно указать его в объявлении VIRTUAL, а имя массива V должно остаться в объявлении типа, но без указания списка границ.

A и V передаются в подпрограмму ABC как массивы, а не как элементы массива, поэтому параметры подпрограммы, соответствующие параметрам A и V, должны быть преобразованы в виртуальные массивы.

Пример трансляции преобразованной программы приведен ниже.

```
FORTRAN IV   B02.00   THU 01-MAY-84 00:41:25          PAGE 001
0001        VIRTUAL A(1000,20), V(1000)
0002        INTEGER*2 V
0003        DO 5, I=1,1000
0004   5    V(I)=0
0005        CALL ABC(A,V,1000,20)
0006        WRITE(2,*)(A(I,1),I=1,1000)
0007        END
FORTRAN IV   STORAGE MAP FOR PROGRAM UNIT .MAIN.
LOCAL VARIABLES, .RSECT $DATA, SIZE = 000006 (     3. WORDS)
NAME TYPE OFFSET  NAME  TYPE  OFFSET    NAME  TYPE  OFFSET
I     I*2   000000
VIRTUAL ARRAYS, TOTAL SIZE = 00240200  (   41024. WORDS)

NAME     TYPE    OFFSET       --------SIZE-------- DIMENSIONS
A       R*4 VEC  00000000     00234200 (  40000.) (1000,20)
V       I*2      00234200     00003720 (   1000.) (1000)


SUBROUTINES,FUNCTIONS,STATAMENTANDPROCEDURE-DEFINEDFUNCTIONS:
NAME  TYPE  NAME   TYPE   NAME  TYPE  NAME  TYPE  NAME  TYPE
ABC     R*4
FORTRAN IV       B02.00   THU  01-MAY-84  00:41:27  PAGE 001
0001        SUBROUTINE ABC(X,Y,N,M)
0002        VIRTUAL Y(N), X(N,M)
0003        INTEGER*2 Y
0004        DO 10, I=1,N
0005   10   X(I,1)=Y(I)
0006        RETURN
0007        END
FORTRAN IV    STORAGE MAP FOR PROGRAM UNIT ABC
LOCAL VARIABLES, .PSECT $DATA, SIZE = 000012 (     5. WORDS)
NAME  TYPE OFFSET  NAME  TYPE  OFFSET   NAME   TYPE  OFFSET
I      I*2  000010  M      I*2 @ 000004   N      I*2 @ 000006
VIRTUAL ARRAYS, TOTAL SIZE = 00000000 (     0. WORDS)

NAME       TYPE      OFFSET    --------SIZE-------- DIMENSIONS
X        R*4    @    000000     ****   (  ****  ) (N,M)
Y        I*2    @    000002     ****   (  ****  ) (N)
```

3.11. Буферизация ввода-вывода

ФОРТРАН выводит информацию в последовательные файлы чаще, чем на терминал, используя для этой цели буфер на 512 символов. Вывод осуществляется по мере заполнения буфера или по закрытию файла. Это особенно заметно при выводе на построчно-печатающее устройство.

Для вывода текущего буфера на построчно-печатающее устройство необходимо закрыть файл, а затем вновь открыть его, используя при этом оператор REWIND.

4. СООБЩЕНИЯ

Транслятор с языка ФОРТРАН, обрабатывая исходную программу, осуществляет предварительный, синтаксический и семантический контроль.

Сообщения предварительного контроля появляются в поле исходной программы листинга непосредственно после предложения, к которому они относятся, при этом трансляция программы не прекращается.

Сообщения предварительного контроля имеют вид

*****<Y>

где Y — буква латинского алфавита, используемая для обозначения ошибки.

Список ошибок предварительного контроля приведен в табл. 10.

Сообщения синтаксического и семантического контроля появляются в листинге программы в поле диагностики и имеют вид:

IN LINE NNNN , ERROR: SSSS

или

IN LINE NNNN , WARNING: SSSS

где NNNN — номер внутренней последовательности предложения;

ERROR — сообщение об ошибке;

WARNING — предупреждение об ошибке (выводится при задании переключателя /W);

SSSS — текст сообщения.

Отметка **** в тексте сообщения означает, что в данном месте текста будет указано имя переменной, массива или метка.

На этапе синтаксического и семантического контроля могут возникнуть неустранимые ошибки.

Сообщения о неустранимых ошибках выводятся на терминал. Появление этих ошибок вызывает прекращение трансляции программы.

Неустранимые ошибки возникают в случае неработоспособности устройства, неправильной подачи команды или если оператор сложен для обработки. Текущая команда или оператор игнорируется, трансляция прекращается.

Сообщения о неустранимых ошибках имеют вид :

FORTRAN—F—SSSS

где SSSS — текст сообщения.

По окончании трансляции на терминал выводится сообщение:

?FORTRAN—I—[S] ERRORS:N, WARNING:M

где S — имя программного модуля;

N — число ошибок;

M — число сообщений, предупреждающих об ошибках.

Таблица 10

Обозначение	Вид ошибки	Причина ошибки	Действие
1	2	3	4
B	W	В позициях 1—5 строки — продолжения встретился, символ, отличный от пробела	Символы в этих позициях игнорируются
C	W	Неправильное продолжение: комментарий или начальная строка первого предложения является строкой — продолжением	Строка игнорируется
E	W	Отсутствует заключительная строка	Модуль транслируется
H	ER	Константа Холлерита содержит более 255 символов	Константа игнорируется
I	W	Использован символ, не входящий в алфавит языка и не содержащийся в текстовой константе или в комментарии	Символ игнорируется
K	W	Недопустимый символ в метке предложения	Метка игнорируется
L	W	Строка содержит более 80 символов (включая символы пробела и табуляции)	Строка усекается
M	W	Многократное определение метки	Повторные метки игнорируются
P	ER	Предложение содержит непарные скобки	Предложение игнорируется
S	ER	Синтаксическая ошибка в записи оператора (например, многочисленные знаки равенства)	Оператор игнорируется
U	ER	Недопустимый формат предложения	Предложение игнорируется

ПРИМЕЧАНИЕ. В графе «Вид ошибки» ER обозначает ошибку, W — предупреждение об ошибке.

4.1. Сообщения синтаксического и семантического контроля

ACCESS='DIRECT' REQUIRES FORM=='UNFORMATTED'

Причина. Ключевое слово FORM='FORMATTED' было указано для файла прямого доступа. ФОРТРАН обеспечивает только бесформатный ввод-вывод прямого доступа

Действие. Согласовать формат файла и метод доступа

ADJUSTABLE DIMENSIONS ILLEGAL FOR ARRAY ****

Причина. Имя массива или имена переменных, представляющих регулируемые размеры, не указаны в подпрограмме как формальные параметры типа целый

Действие. Массиву присваивается единичная размерность. Исправить исходную программу

ARRAY **** EXCEEDS MAXIMUM SIZE
или ARRAY EXCEEDS MAXIMUM SIZE

Причина. Недостаточно памяти для указанного массива (первое сообщение) или для всех массивов в программе

Действие. Проверить объявления массивов и уменьшить общие требования к памяти для массивов

ARRAY **** HAS TOO MANY DIMENSIONS

Причина. Массив имеет более семи измерений

Действие. Исправить объявление массива и предложения, связанные с ним

**** ATTEMPTS TO EXTEND COMMON BLOCK BACKWARDS

Причина. Попытка расширения блока COMMON влево объявлением эквивалентности

Действие. Согласовать объявления общих объектов и эквивалентности

COMMON BLOCK EXCEEDS MAXIMUM SIZE

Причина. Попытка предоставить для блока COMMON больше места, чем физически возможно (более 32K слов)

Действие. Исправить предложение

CONSTANT IN FORMAT STATEMENT NOT IN RANGE

Причина. Целая константа в объявлении формата находится вне допустимых пределов (1—255)

Действие. Исправить объявление формата

DANGLING OPERATOR

Причина. В выражении пропущен операнд

Действие. Исправить неправильное выражение

DEFECTIVE DOTTED KEYWORD

Причина. Неправильная запись знака операции отношения

Действие. Исправить запись отношения

DEFINE FILE MODE MUST BE 'U'

Причина. Третий параметр в скобках в объявлении DEFINE FILE не 'U'

Действие. Исправить параметр в объявлении

DO TERMINATOR **** PRECEDES DO STATEMENT

Причина. Отсутствует закрывающий оператор тела цикла

Действие. Поместить закрывающий оператор тела цикла за оператором DO

EXPECTING LEFT PARENTHESIS AFTER ****

Причина. Отсутствует левая скобка за именем массива или именем функции

Действие. Исправить предложение

EXPECTING LEFT PARENTHESIS AFTER SUBPROGRAM NAME

Причина. Не указан список параметров за именем SUBROUTINE или FUNCTION

Действие. Проверить предложение: нет ли опечатки, и не используется ли одно и то же имя для переменной и подпрограммы

EXTRA CHARACTERS AT END OF STATEMENT

Причина. За допустимым предложением в строке следуют дополнительные символы

Действие. Проверить: не пропущена ли запятая, нет ли лишних символов в конце предложения и не ошибочен ли символ в шестой позиции следующей строки

FLOATING CONSTANT NOT IN RANGE

Причина. Значение константы с плавающей точкой в выражении близко к нулю

Действие. Использовать, если возможно, константу 0.0

ILLEGAL ADJACENT OPERATOR

Причина. Два знака операции расположены друг за другом

Действие. Исправить неправильное выражение

ILLEGAL CHARACTERS IN EXPRESSION

Причина. Недопустимый символ в выражении

Действие. Исправить выражение

ILLEGAL DO TERMINATOR ORDERING AT LABEL ****

Причина. Неправильная организация гнезда в операторе цикла

Действие. Организовать правильно гнездо

ILLEGAL DO TERMINATOR STATEMENT ****

Причина. Недопустимый закрывающий оператор тела цикла

Действие. Проверить, чтобы закрывающий оператор тела цикла не был оператором GOTO, арифметическим IF, RETURN, другим оператором DO или логическим IF, содержащим один из этих операторов

ILLEGAL ELEMENT IN I/O LIST

Причина. Список ввода-вывода содержит синтаксическую ошибку

Действие. Исправить список ввода-вывода

ILLEGAL ENCODE/DECODE FORMAT SPECIFIER

Причина. В операторе ENCODE или DECODE второй параметр в скобках (указатель формата) не является меткой объявления формата или именем массива, содержащим спецификацию формата

Действие. Исправить данный параметр

ILLEGAL ENCODE/DECODE LENGTH EXPRESSION

Причина. В операторе ENCODE или DECODE первый параметр в скобках (число преобразуемых символов) не является целым выражением

Действие. Исправить данный параметр

ILLEGAL ENCODE/DECODE TARGET

Причина. В операторе ENCODE или DECODE третий параметр в скобках не является именем массива, элементом массива или именем переменной

Действие. Исправить данный параметр

ILLEGAL INITIAL VALUE EXPRESSION IN DO STATEMENT

Причина. В операторе DO недопустимое значение начального параметра

Действие. Исправить значение начального параметра

ILLEGAL STATEMENT IN BLOCK DATA

Причина. В модуле-блоке данных обнаружено недопустимое предложение

Действие. Исключить недопустимое предложение из модуля-блока данных

ILLEGAL STATEMENT ON LOGICAL IF

Причина. Оператор, содержащийся в логическом IF, не является допустимым (например, оператор DO или другой логический IF)

Действие. Исправить оператор IF

ILLEGAL SUBSCRIPTS OR SUBPROGRAM ARGUMENT

Причина. Недопустимый элемент в списке параметров подпрограммы или в индексе массива

Действие. Исправить предложение

ILLEGAL TYPE FOR OPERATOR

Причина. В экспоненциальной функции или условном логическом операторе использована переменная недопустимого типа

Действие. Проверить тип переменной в данной функции или операторе

ILLEGAL USAGE OF OR MISSING LEFT PARENTHESIS

Причина. Присутствует недопустимая или отсутствует левая скобка

Действие. Исправить ошибочное выражение

INTEGER OVERFOW

Причина. Целая константа или значение выражения находится вне диапазона от —32767 до +32767

Действие. Изменить константу или выражение так, чтобы его значение находилось в указанном диапазоне

INVALID COMPLEX CONSTANT

Причина. Недопустимый формат комплексной константы

Действие. Исправить формат комплексной константы

INVALID DIMENSIONS FOR ARRAY ****

Причина. Недопустимые измерения массива

Действие. Проверить: чтобы ноль не использовался в списке границ, число измерений должно находиться в диапазоне от 1 до 7

INWALID END= OR ERR= KEYWORD

Причина. Недопустимый формат параметра END= или ERR= в операторе ввода или вывода

Действие. Исправить оператор ввода или вывода

INVALID EQUIVALENCE

Причина. Объявление эквивалентности недопустимо, или оно не согласуется с предыдущим объявлением эквивалентности

Действие. Исправить объявление эквивалентности

INVALID FORMAT SPESIFIER

Причина. Указатель формата не является ни меткой объявления формата, ни именем массива, содержащим спецификацию формата
Действие. Исправить указатель формата

INVALID IMPLICIT RANGE SPECIFIER

Причина. В объявлении типа IMPLICIT обнаружен небуквенный символ или указана последовательность символов в неалфавитном порядке
Действие. Исправить неявное объявление типа

INVALID LOGICAL UNIT

Причина. Неверная ссылка на номер логического устройства
Действие. Исправить номер устройства так, чтобы он был целой переменной или константой в пределах от 1 до 99

INVALID OCTAL CONSTANT

Причина. Значение восьмеричной константы находится вне диапазона (0—177777) или содержит цифры, отличные от 0—7
Действие. Исправить восьмеричную константу

INVALID OPTIONAL LENGTH SPECIFIER

Причина. Недопустимый указатель длины в объявлении типа данных
Действие. Исправить объявление типа данных

INVALID RADIX-50 CONSTANT

Причина. Недопустимый символ в константе RADIX-50
Действие. Исправить константу RADIX-50

INVALID STATEMENT LABEL REFERENCE

Причина. Ссылка на метку неправильной конструкции
Действие. Проверить, чтобы метка состояла из десятичных цифр от одной до пяти, расположенных в первых пяти позициях строки, и не состояла из одних нулей

INVALID PROGRAM NAME

Причина. Символическое имя, использованное в операторе вызова подпрограммы или в ссылке на функцию, недопустимо
Действие. Записать правильно имя

INVALID TARGET FOR ASSIGNMENT

Причина. В арифметическом или логическом операторе присваивания слева от знака равенства указано не имя переменной и не имя элемента массива
Действие. Исправить оператор

INVALID TYPE SPECIFIER

Причина. Недопустимый тип данных
Действие. Исправить тип данных

INVALID USAGE OF SUBROUTINE OR FUNCTION NAME

Причина. Используется имя функции или подпрограммы в объявлении DEMENSION, COMMON, DATA, EQUIVALENCE или в объявлении типа данных
Действие. Исключить имя функции или подпрограммы из данного объявления

INVALID VARIABLE NAME

Причина. Недопустимое имя переменной
Действие. Исправить имя переменной

LABEL ON DECLARATIVE STATEMENT

Причина. Метка предложения недопустима
Действие. Удалить метку

MISSING ASSIGNMENT OPERATOR

Причина. В арифметическом или логическом операторе присваивания пропущен знак равенства или он находится не на своем месте
Действие. Исправить оператор

MISSING COMMA

Причина. Отсутствует запятая
Действие. Исправить предложение

MODE OF EXPRESSION MUST BE INTEGER

Причина. Используется не целая константа или переменная, например в операторе DO начальный параметр, конечный параметр или параметр приращения имеет не целое значение
Действие. Исправить оператор

MISSING COMMA IN OPEN OR CLOSE KEYWORD LIST

Причина. Два ключевых слова в операторе OPEN или CLOSE не разделены запятой
Действие. Исправить оператор

MISSING DELIMITER IN EXPRESSION

Причина. Пропущен знак операции между двумя операндами в выражении
Действие. Исправить выражение

MISSING EXPRESSION

Причина. Пропущено требуемое выражение (например, конечный параметр в операторе DO)
Действие. Скорректировать предложение

MISSING LABEL

Причина. В операторе отсутствует метка или указана синтаксически неверно

Действие. Исправить оператор
MISSING LABEL LIST AFTER COMMA

Причина. В операторе перехода по предписанию отсутствует список меток

Действие. Исправить оператор
MISSING LEFT PARENTHESIS AFTER OPEN OR CLOSE

Причина. В операторе OPEN или CLOSE отсутствует левая скобка перед списком ключевых слов

Действие. Исправить оператор
MISSING OPERATOR AFTER EXPRESSION

Причина. Выражение в операторе не ограничено запятой, правой скобкой или другим оператором

Действие. Исправить оператор
MISSING QUOTATION MARK

Причина. В операторе FIND номер логического устройства и номер записи не разделены апострофом

Действие. Исправить оператор
MISSING RIGHT PARENTHESIS

Причина. Правые скобки находятся не на месте или пропущены

Действие. Исправить предложение
MISSING 'TO' IN ASSING STATEMENT

Причина. Отсутствует 'TO' в операторе ASSIGN

Действие. Исправить оператор
MISSING VALUE FOR KEYWORD IN OPEN OR CLOSE STATEMENT

Причина. Отсутствует значение ключевого слова в операторе OPEN или CLOSE

Действие. Исправить оператор
MISSING VARIABLE

Причина. В предложении пропущена переменная

Действие. Исправить предложение
MISSING VARIABLE OR CONSTANT

Причина. Вместо переменной или константы использован разделитель (запятая, скобки и так далее)

Действие. Проверить формат предложения и исправить его
MODE OF EXPRESSION MUST BE INTEGER

Причина. Выражение или его значение не принадлежит к типу целый

Действие. Исправить выражение
MODES OF VARIABLE **** AND DATA ITEM DIFFER

Причина. В объявлении DATA тип переменной и связанный с ним элемент списка не согласованы

Действие. Согласовать тип переменной с элементом списка
MULTIPLE DECLARATION OR VARIABLE ****

Причина. Многократное описание массива или объявление типа переменной

Действие. Оставить одно объявление типа переменной или одно описание массива
MULTIPLE DECLARATION OF OPEN OR CLOSE KEYWORD

Причина. Многократное определение ключевого слова в операторе OPEN или CLOSE

Действие. Оставить одно определение ключевого слова
OPEN OR CLOSE KEYWORD VALUE MUST BE QUOTED STRING

Причина. Значение ключевого слова в операторе OPEN или CLOSE не заключено в апострофы

Действие. Заключить в апострофы значение ключевого слова
OPEN OR CLOSE STATEMENT REQUIRES UNIT=SPECIFIER

Причина. Отсутствует ключевое слово UNIT в операторе OPEN или CLOSE

Действие. Исправить оператор
PARENTHESES NESTED TOO DEEPLY

Причина. В объявлении формата описатели полей, разделители полей или основные группы, заключенные в скобки, превышают восьмикратное вложение

Действие. Проверить спецификацию формата
PROGRAM OR BLOCK DATA STATEMENT MUST BE FIRST

Причина. Заголовок головного модуля или заголовок блока данных не является первым предложением модуля

Действие. Исправить заголовок
P—SCALE FACTOR NOT IN RANGE —127 TO +127

Причина. Масштабный множитель в спецификации формата находится вне диапазона (от —127 до +127)

Действие. Исправить масштабный множитель
REFERENCE TO INCORRECT TYPE OF LABEL ****

Причина. Логически неверное использование ссылки на метку

Действие. Исправить программу

REFERENCE TO UNDEFINED STATEMENT LABEL

Причина. Ссылка на метку, неопределенную в данном модуле

Действие. Исправить программу

STATEMENT MUST BE UNLABELED

Причина. Объявление внутренней функции имеет метку

Действие. Удалить метку

STATEMENT TOO COMPLEX

Причина. Объявление внутренней функции имеет более десяти параметров или является слишком длинным для его обработки

Действие. Уменьшить список формальных параметров до десяти или разбить объявление на два или более

SUBROUTINE OR FUNCTION STATEMENT MUST BE FIRST

Причина. Заголовок SUBROUTINE, FUNCTION или BLOCK DATA не является первым в программном модуле

Действие. Исправить модуль

SUBSCRIPT OF ARRAY **** NOT N RANGE

Причина. Индекс массива принимает недопустимое значение

Действие. Исправить индекс

SYNTAX ERROR

Причина. Недопустимый формат

Действие. Исправить ошибочное предложение

SYNTAX ERROR IN INTEGER OR FLOATING CONSTANT

Причина. Недопустимый формат целой, вещественной, двойной точности или комплексной константы

Действие. Исправить данную константу

SYNTAX ERROR IN LABEL LIST

Причина. В списке меток содержится ошибка

Действие. Проверить формат списка меток и убедиться, что каждая метка является меткой выполняемого предложения. Исправить оператор

TARGET MUST BE ARRAY

Причина. В операторе ENCODE или DECODE третий параметр в скобках является именем виртуального массива или элемента виртуального массива

Действие. Исправить данный параметр

UNARY OPERATOR HAS TOO MANY OPERANDS

Причина. Указано два или более операндов для операции, использующей только один операнд

Действие. Проверить ошибки печати в операторе, пропущенные операции и пропущенные скобки. Исправить оператор

UNLABELED FORMAT STATEMENT

Причина. Объявление формата не имеет метки

Действие. Пометить объявление формата

UNRECOGNIZED KEYWORD IN OPEN OR CLOSE STATEMENT

Причина. Недопустимое ключевое слово в операторе OPEN или GLOSE

Действие. Проверить нет ли опечаток в операторе OPEN или CLOSE и исправить их

UNRECOGNIZED VALUE FOR OPEN OR CLOSE KEYWORD

Причина. Ключевое слово в операторе OPEN или CLOSE указано с недопустимым значением. Например, DISROSE='SURE'

Действие. Исправить оператор

USAGE OF VARIABLE **** INVALID

Причина. Использование переменной недопустимо в данном предложении. Например, недопустимо использование в качестве параметра в объявлении эквивалентности переменной из блока COMMON, массива переменных или формального параметра, или недопустимо размещение формального параметра или внешнего имени в блоке COMMON

Действие. Исправить предложение

VALUE OF CONSTANT NOT IN RANGE

Причина. Значение константы превышает допустимый диапазон

Действие. Исправить предложение

VARIABLE **** INVALID IN ADJUSTABLE DIMENSION

Причина. В списке формальных параметров, представляющих регулируемые массивы, встречается имя переменной нецелого типа

Действие. Исправить программу

WRONG NUMBER OF OPERANDS FOR BINARY OPERATOR

Причина. В операторе отсутствует второй необходимый операнд

Действие. Проверить ошибки печати и исправить оператор
WRONG NUMBER OF SUBSCRIPTS FOR ARRAY ****
Причина. Число индексных выражений не соответствует объявленному числу измерений массива
Действие. Исправить ссылку на массив

4.1.1. Сообщения, предупреждающие об ошибках

LOOP ENTRY AT LABEL ****
Причина. Передача управления осуществляется извне в тело цикла, содержащее указанную метку
Действие. Исправить передачу управления
NON—STANDARD STATEMENT ORDERING
Причина. Нарушен порядок следования предложений
Действие. Исправить порядок следования предложений
POSSIBLE MODIFICATION OF ****
Причина. Недопустимая модификация переменной. Например, управляющая переменная изменяется операторами внутри цикла или осуществляется передача управления на метку внутри тела цикла из оператора, находящегося вне тела цикла
Действие. Исправить программу
VARIABLE **** IS NOT WORD ALIGNED
Причина. Переменная или массив, которые не относятся к типу LOGIKAL*1, помещены в объявлении эквивалентности с переменными или массивами типа LOGICAL*1 или в блоке COMMON после переменной или массива типа LOGICAL*1
Действие. Выравнить переменную или массив по границе слова VARIABLE **** NAME EXCEEDS SIX CHARACTERS
Причина. Имя переменной **** превышает шесть символов
Действие. Исправить имя

4.1.2. Сообщения о неустранимых ошибках

?FORTRAN—F—CODE GENERATION STACK OVERFLOW
Причина. Переполнение стека. Оператор программы сложен для обработки
Действие. Упростить сложный оператор
?FORTRAN—F—COMPILER FATAL ERROR, ANALYSIS FOLLOWS
Причина. Произошел сбой транслятора. Далее следует краткое сообщение о характере сбоя

Действие. Если сообщение содержит рекомендации по устранению сбоя, следует воспользоваться ими
?FORTRAN—F—CONSTANT SUBSCRIPT STACK OVERFLOW
Причина. Программа имеет оператор с большим количеством индексных констант
Действие. Упростить оператор
?FORTRAN—F—DEVICE FULL
Причина. На томе, используемом для вывода объектного файла или файла листинга, недостаточно свободного места или полностью заполнен справочник
Действие. Освободить место на томе или использовать другой том
?FORTRAN—F—DYNAMIC MEMORY OVERFLOW
Причина. Недостаточно оперативной памяти для трансляции программного модуля
Действие. Разбить программный модуль на подпрограммы или транслировать существующий модуль на ЭВМ с большей памятью
?FORTRAN—F—ERROR READING SOURCE FILE
Причина. Ошибка при считывании входного файла
Действие. Проверить готовность и исправность оборудования. Повторить трансляцию
?FORTRAN—F—ERROR WRITING LISTING FILE
Причина. Ошибка при записи файла листинга
Действие. Проверить готовность и исправность оборудования. Проверить диск на плохие блоки. Повторить трансляцию
?FORTRAN—F—ERROR WRITING OBJECT FILE
Причина. Ошибка при записи объектного файла
Действие. Проверить готовность и исправность оборудования. Проверить диск на плохие блоки. Повторить трансляцию
?FORTRAN—F—FILE NOT FOUND
Причина. Входной файл, указанный в командной строке, не найден
Действие. Проверить, существует ли файл с указанным именем. Ввести правильную командную строку
?FORTRAN—F—HELP NOT FOUND
Причина. На томе отсутствует файл FORTRA.HLP
Действие. Скопировать файл FORTRA.HLP с дистрибутивного тома, если требуется вспомогательная информация
?FORTRAN—F—ILLEGAL VALUE FOR /X SWITCH

Причина. В командной строке указан недопустимый параметр переключателя /X

Действие. Ввести правильную командную строку

?FORTRAN—F—ILLEGAL COMMAND

Причина. Недопустимый формат командной строки

Действие. Ввести правильную командную строку

?FORTRAN—F—ILLEGAL DEVICE

Причина. В командной строке обнаружено недопустимое имя устройства

Действие. Ввести правильную командную строку

?FORTRAN—F—OPTIMIZER STACK OVERFLOW

Причина. Оператор программы сложен для обработки или в программном модуле много общих подвыражений

Действие. Упростить сложные операторы

?FORTRAN—F—SUBEXPRESSION STACK OVERFLOW

Причина. При трансляции обнаружен оператор, который может вызвать переполнение стека во время выполнения программы

Действие. Упростить сложный оператор

?FORTRAN—F—UNKNOWN SWITCH — /X

Причина. В командной строке обнаружен недопустимый переключатель

Действие. Ввести правильную командную строку

ПРИЛОЖЕНИЕ 1

КОДЫ СИМВОЛОВ

В данном приложении приведены наборы символов КОИ-7 (табл. 1) и RADIX-50 (табл. 2) и их коды, а также позиционное значение кода RADIX-50 (табл. 3).

Таблица 1

КОДЫ КОИ—7

Код	Символ	Код	Символ	Код	Символ	Код	Символ
1	2	3	4	5	6	7	8
000	ПУС	040	ПРОБЕЛ	100	@	140	Ю
001		041	!	101	А	141	А
002		042	"	102	В	142	Б
003		043	#	103	С	143	Ц
004		044	$	104	D	144	Д
005		045	%	105	E	145	Е
006		046	&	106	F	146	Ф
007		047	'	107	G	147	Г
010	ВШ	050	(	110	H	150	Х
011	ГТ	051	)	111	I	151	И
012	ПС	052	*	112	J	152	Й
013		053	+	113	K	153	К
014	ПФ	054	,	114	L	154	Л
015	ВК	055	—	115	M	155	М
016	ВР	056	.	116	N	156	Н
017	НР	057	/	117	O	157	О
020		060	0	120	P	160	П
021	КЛ	061	1	121	Q	161	Я
022	ЧЛ	062	2	122	R	162	Р
023		063	3	123	S	163	С
024		064	4	124	T	164	Т
025		065	5	125	U	165	У
026		066	6	126	V	166	Ж
027		067	7	127	W	167	В
030		070	8	130	X	170	Ь
031		071	9	131	Y	171	Ы
032		072	:	132	Z	172	З
033		073	;	133	[	173	Ш
034		074	<	134	\	174	Э
035		075	=	135	]	175	Щ
036		076	>	136	^	176	Ч
037		077	?	137	—	177	ЗБ

Таблица 2

КОДЫ СИМВОЛОВ RADIX—50

Символ	Код КОИ 7	Код RADIX 50
1	2	3
ПРОБЕЛ	40	0
A—Z	101—132	1—32
$	44	33
.	56	34
0—9	60—71	36—47
Не используется	—	35

Таблица 3

КОДЫ ПОЗИЦИОНИРОВАННЫХ СИМВОЛОВ RADIX—50

Символ	Код одиночного или первого символа	Код второго символа	Код третьего символа
1	2	3	4
A	003100	000050	000001
B	006200	000120	000002
C	011300	000170	000003
D	014400	000240	000004
E	017500	000310	000005
F	022600	000360	000006
G	025700	000430	000007
H	031000	000500	000010
I	034100	000550	000011
J	037200	000620	000012
K	042300	000670	000013
L	045400	000740	000014
M	050500	001010	000015
N	053600	001060	000016
O	056700	001130	000017
P	062000	001200	000020
Q	065100	001250	000021
R	070200	001320	000022
S	073300	001370	000023
T	076400	001440	000024
U	101500	001510	000025
V	104600	001560	000026
W	107700	001630	000027
X	113000	001700	000030
Y	116100	001750	000031
Z	121200	002020	000032
$	124300	002070	000033
.	127400	002140	000034
Не исп.	132500	002210	000035
0	135600	002260	000036
1	140700	002330	000037
2	144000	002400	000040
3	147100	002450	000041
4	152200	002520	000042
5	155300	002570	000043
6	160400	002640	000044
7	163500	002710	000045
8	166600	002760	000046
9	171700	003030	000047

Используя табл. 3 можно быстро вычислить значение константы RADIX-50. Например, значение константы RADIX-50 X2B вычисляется следующим образом (вычисление выполняется в восьмеричной системе):

X=113000
2=002400
B=000002
X2B=115402

ПРИЛОЖЕНИЕ 2

ФОРМАТЫ ДАННЫХ

В данном приложении приведено представление различных типов данных, допустимых в языке ФОРТРАН [5].

1. Формат данных типа INTEGER

Данные типа INTEGER, INTEGER*2 (целые данные) записываются в памяти ЭВМ в одно слово в виде двоичного целого числа (рис. 1). Отрицательные целые числа записываются в виде дополнительного кода.

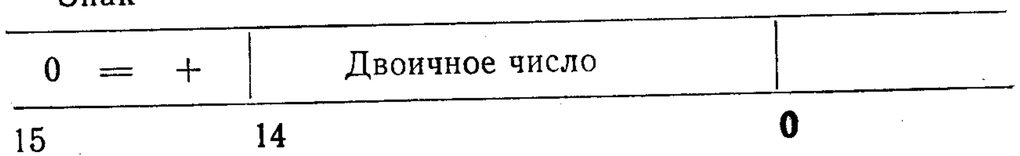


Рис. 1

Диапазон представления целых чисел от —32768 до 32767.

Данные INTEGER*4 и INTEGER записываются в памяти в два слова, если указан переключатель /T или /I4 в командной строке, но значащим является только слово с младшим адресом.

Примеры:

+22=000026(8)
— 7=177771

2. Формат данных типа REAL

Данные типа REAL, REAL*4 (вещественные данные) записываются в памяти ЭВМ в два слова в виде двоичного числа с плавающей точкой с обычной точностью (рис. 2).

Порядок числа с плавающей точкой записывается с избытком (P=P+200), т. е. порядки от —128 до 127 представлены двоичными эквивалентами чисел от 0 до 255 (от 0 до 377 восьмеричных).

Мантисса числа с плавающей точкой является нормализованной (старший разряд имеет единичное значение, за

исключением мантиссы нулевого числа), и поэтому она записывается без старшего разряда, который избыточен.

Число ноль имеет нулевой порядок и нулевую мантиссу. Диапазон представления чисел с плавающей точкой от 0.29*10**(—38) до 0.17*10**39. Точность представления мантиссы 23+1=24 бит, что соответствует приблизительно 7 десятичным цифрам.

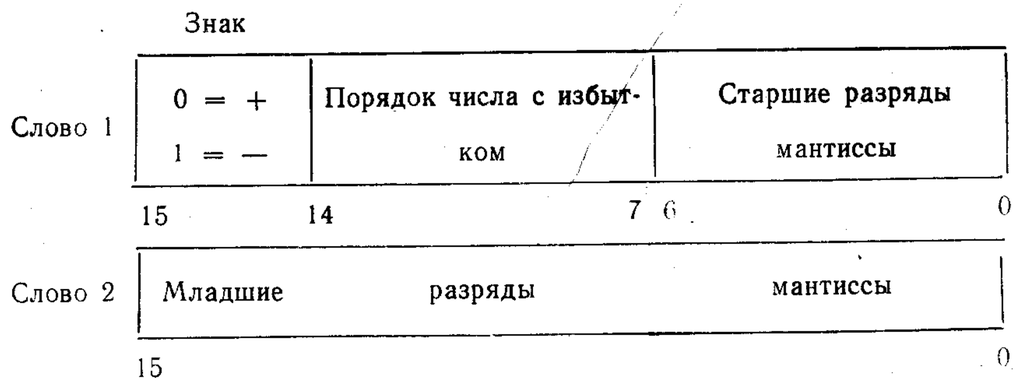


Рис. 2

Примеры:

Число +1.0 записывается в две последовательные ячейки в виде 40200,0.

Число —5 записывается в виде 140640,0

3. Формат данных типа DOUBLE PRECISION

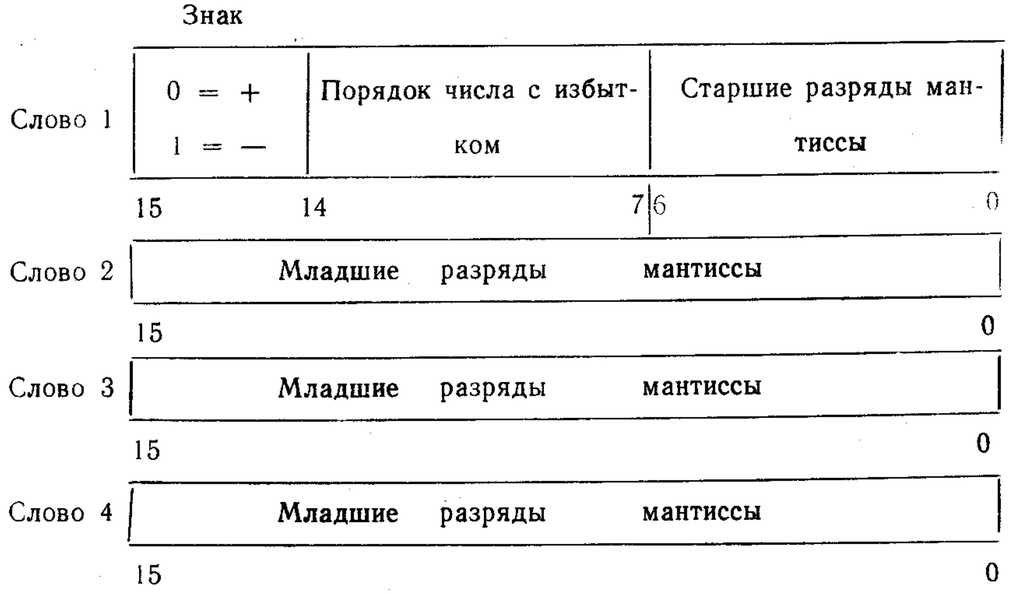


Рис. 3

Данные типа DOUBLE PRECISION, REAL*8 (данные двойной точности) записываются в памяти ЭВМ в четыре слова в виде двоичного числа с плавающей точкой с двойной точностью (рис. 3).

Точность представления мантиссы 56 бит, что соответствует приблизительно 17 десятичным цифрам.

4. Формат данных типа COMPLEX

Данные типа COMPLEX, COMPLEX*8 (комплексные данные) записываются в памяти ЭВМ в четыре слова в виде пары чисел с плавающей точкой с обычной точностью (рис. 4).

Первое число представляет действительную часть комплексного числа, второе — мнимую часть.

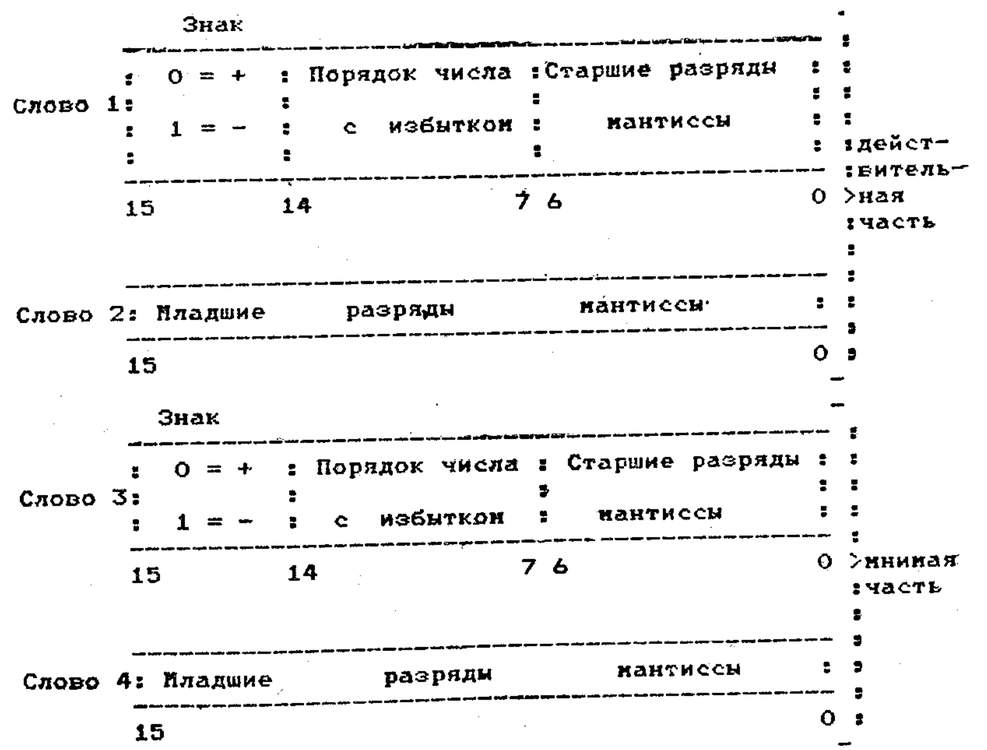


Рис. 4

5. Формат данных типа LOGICAL

Данные типа LOGICAL, LOGICAL*4 (логические данные) записываются в памяти ЭВМ в два слова (рис. 5) в виде логической константы «истина» (код 377 в младшем байте) и «ложь» (код 000 в младшем байте).

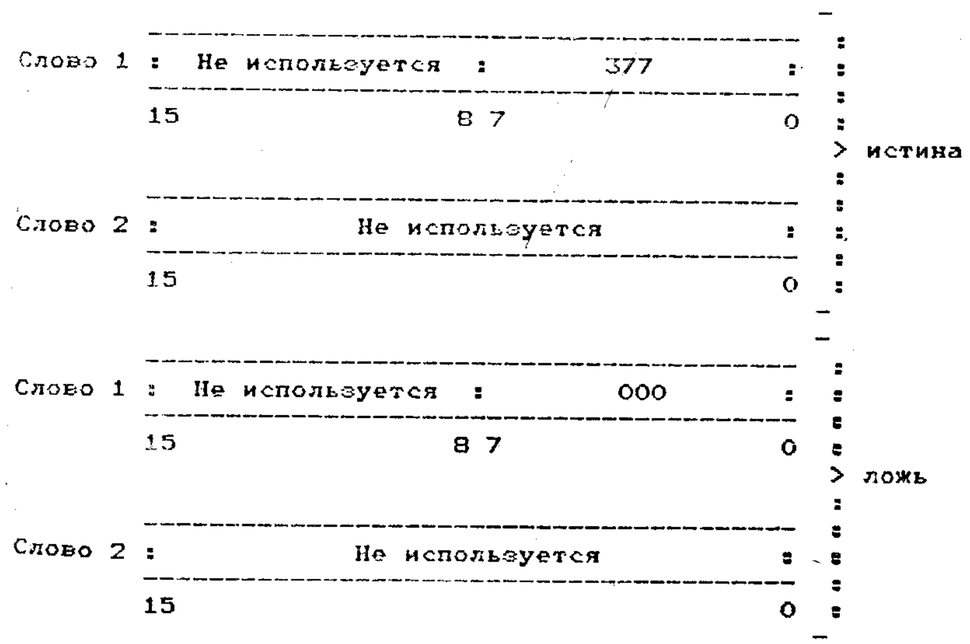


Рис. 5

В логических выражениях любое ненулевое значение младшего байта рассматривается как логическое значение «истина».

Данные LOGICAL*1 записываются аналогично байтовым данным (рис. 6). Любое ненулевое значение рассматривается как логическое значение «истина», нулевое значение — «ложь».

6. Формат данных типа BYTE

Данные типа BYTE (байтовые данные) записываются в памяти ЭВМ в последовательных байтах (рис. 6).

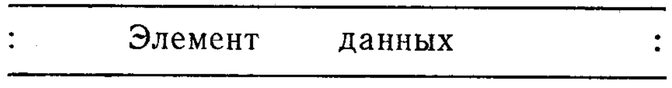


Рис. 6

В данном формате записываются восьмеричные числа от —128 до +127 и единичные символы.

7. Формат текстовых данных

Текстовые данные, представленные в виде константы Холлерита и буквенно-цифрового литерала, записываются в памяти ЭВМ в коде КОИ-7 в последовательных байтах (рис. 7). Каждый символ текстовой константы занимает один байт. Значение текстовой константы дополняется пробелом, если число символов нечетно.

Константы RADIX-50 записываются в памяти ЭВМ по три символа в одно слово:

значение константы RADIX-50 = ((C1*50)+C2)*50+C3

где C1, C2, C3 — код символа RADIX-50 (см. приложение 1);

50 — восьмеричная константа.

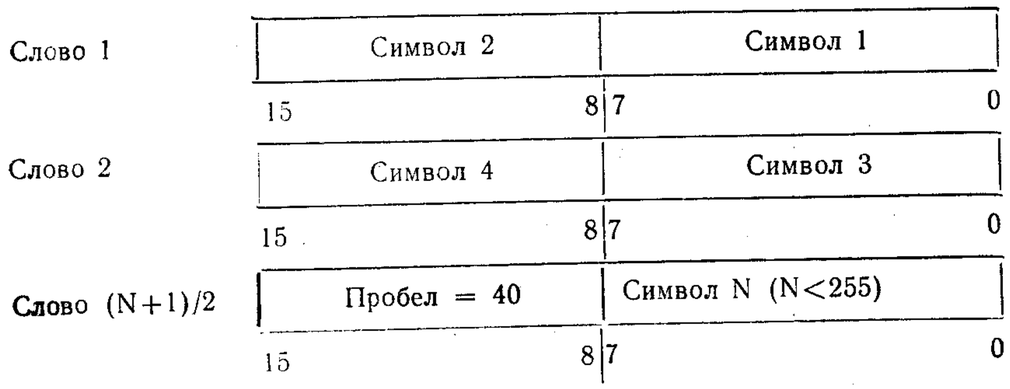


Рис. 7

ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Операционная система ФОДОС-2
   АССЕМБЛЕР
   Руководство программиста
2. Операционная система ФОДОС-2
   Редактор связей
   Руководство оператора
3. Операционная система ФОДОС-2
   Командный язык системы
4. ФОРТРАН/ФОДОС-2
   Библиотека ФОРТРАНа
   Руководство программиста
5. ФОРТРАН/ФОДОС-2
   Описание языка

ФОРТРАН/ФОДОС-2. БИБЛИОТЕКА ФОРТРАНа
РУКОВОДСТВО ПРОГРАММИСТА
1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

Библиотека ФОРТРАНа (файл FORLIB.OBJ) является библиотекой объектных модулей, используемых программами на языке ФОРТРАН.

Библиотека ФОРТРАНа состоит из:

— подпрограмм внешних функций языка ФОРТРАН;
— вспомогательных подпрограмм (ASSIGN, CLOSE, DATE и т. д.);
— подпрограмм выполнения различных операций ввода-вывода;
— подпрограмм обработки ошибок;
— подпрограмм для создания объектных модулей.

Необходимые модули библиотеки включаются в загрузочный модуль программы пользователя с помощью программы «Редактор связей» [1]. В командной строке редактора связей библиотека ФОРТРАНа может быть задана явно в виде

уст: FORLIB.OBJ

где уст: — устройство, на котором находится файл FORLIB.OBJ или неявно по переключателю /F.

Если для таблицы имен библиотеки ФОРТРАНа недостаточно оперативной памяти, следует использовать переключатель /S в командной строке редактора связей. В этом случае для таблицы имен резервируется наибольший возможный участок памяти за счет того, что каталог библиотеки остается на системном устройстве, но не является резидентным. Т. к. библиотека ФОРТРАНа является составной частью транслятора с ФОРТРАНа, то условия применения у нее такие же как и для транслятора [2].

2. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

2.1. Структура библиотеки ФОРТРАНа

Структура библиотеки ФОРТРАНа приведена на рисунке.

: Заголовок библиотеки :
: Таблица точек входа :
: Объектные модули :
: Конец библиотеки :

Заголовок библиотеки состоит из 17-ти десятичных слов, описывающих текущее состояние библиотеки (версия программы библиотекарь, дата и время создания библиотеки и ее корректировки, относительный адрес таблицы точек входа, допустимое и используемое число точек входа; указано место, куда будет вставляться следующий модуль).

Таблица точек входа состоит из описания точек входа по четыре слова на каждую точку.

Объектный модуль состоит из последовательности записей.

Блок конца библиотеки содержит признак окончания библиотеки.

2.2. Объектный код

Как правило для трансляции каждого оператора ФОРТРАНа необходима определенная последовательность машинных команд. Например, после выполнения тела оператора цикла управляющая переменная увеличивается на значение, представленное соответствующим параметром приращения, выполняется шаг оператора цикла. Для размещения элементов массива со многими измерениями, инициации операций ввода-вывода или моделирования операций с плавающей точкой в библиотеке ФОРТРАНа есть стандартные последовательности машинных команд. Транслятор с языка ФОРТРАН выбирает определенную комбинацию этих последовательностей для выполнения оттранслированной программы.

Транслятор с ФОРТРАНа выдает объектный код двух видов: поточный и нанизанный.

Поточный код — это объектный код программы на языке АССЕМБЛЕР, зависящий от набора команд процессора (базовый набор с использованием команд расширенной арифметики — EIS или базовый набор с использованием команд расширенной арифметики и плавающей запятой — FIS).

Для поточного кода транслятор с ФОРТРАНа обращается к библиотеке ФОРТРАНа только в том случае, если нельзя выполнить данную функцию небольшой последовательностью команд.

Нанизанный код — это код, генерируемый в виде обращения к программам библиотеки ФОРТРАНа. Нанизанный код использует только базовый набор команд процессора.

Транслятор с ФОРТРАНа обращается к библиотечной последовательности команд с помощью генерирования слова, содержащего адрес первой команды в последовательности, за адресом следует информация, на основании которой подпрограмма должна работать. Например, в случае оператора DO за генерированным словом следуют адреса параметров цикла и адрес начала цикла.

Регистр общего назначения R4 используется для обращения к подпрограммам — последовательностям команд библиотеки ФОРТРАН. Последняя выполняемая команда каждой последовательности команд есть JMP 0(R4)+, которая передает управление следующей последовательности команд.

Мнемоника глобальных имен, используемых в качестве имен подпрограмм библиотеки ФОРТРАНа, содержит от четырех до шести символов. Первые два символа определяют операцию. Третий символ определяет тип данных, над которыми осуществляется операция (INTEGER, REAL, COMPLEX). Четвертый символ — всегда $. Пятый и шестой, если они присутствуют, определяют источник и приемник информации. Источником может быть ячейка памяти, стек, регистры общего назначения или последовательные ячейки памяти, обращение к которым осуществляется через регистр общего назначения R4. Приемником информации может быть ячейка памяти, стек или последовательные ячейки памяти, обращение к которым осуществляется через регистр R4.

К функциям подпрограмм библиотеки ФОРТРАНа относятся осуществление арифметических операций, сравнение переменных по величине, определение типа переменных, вычисление индекса в многомерных массивах, преобразование типа данных и передача управления другой подпрограмме. Имеется специальная подпрограмма нумерации строк внутренней последовательности, подпрограмма управления вызовом подпрограмм, подпрограмма управления стеком. Имеются несколько подпрограмм для реализации операторов языка ФОРТРАН таких, как PAUSE, STOP, инициализации периферийных устройств ввода-вывода и обмена данными. Например, при трансляции следующей программы

```
0001      DIMENSION RARRAY(10, 10)
0002      DATA N/4/, I/2/
0003      I=(3*2—5)+I
0004      J=(I+100)*(N**2)
```



```
0005      A=2.0
0006      RARRAY(2, 1)=RARRAY(1, 1)+A
0007      END
```

генерируется нанизанный объектный код в виде обращения к подпрограммам библиотеки ФОРТРАНа.

```
STATEMENT #0003
000006 LSN$
000012 ICI$M          #000003
                      $DATA+#000622
STATEMENT #0004
000016 ISN$
000020 MOI$MS         $DATA+#000622
000024 ADI$IS         #000144
000030 MOI$MS         $DATA+#000620
000034 MUI$MS         $DATA+#000620
000040 MUI$SS
000042 MOI$SM         $DATA+#000624
STATEMENT #0005
000046 ISN$
000050 MOF$IM         #040400  $DATA+#000626
STATEMENT #0006
000056 ISN$
000060 MOF$MM         $DATA+#000626   $DATA+#000004
000066 ADF$MM         $DATA+#000000   $DATA+#000004
STATEMENT #0007
000074 ISN$
000076 RET$
```

2.3. Вспомогательные подпрограммы библиотеки ФОРТРАНа

В дополнение к внешним функциям языка ФОРТРАН библиотека ФОРТРАНа содержит следующие подпрограммы:

ASSIGN — устанавливает соответствие между спецификацией файла и номером логического устройства;

GLOSE — закрывает файл на указанном логическом устройстве;

DATE — определяет текущую дату в виде последовательности символов;

IDATE — определяет текущую дату в виде трех целых чисел;

EXIT — заканчивает выполнение программы пользователя и передает управление монитору;

USEREX — задает имя подпрограммы завершения;

RANDU — генерирует случайное число на интервале от 0 до 1;

SETERR — изменяет реакцию библиотеки ФОРТРАНа по ошибке;
ERRTST — контролирует появление ошибки, указанной пользователем;
ERRSNS — регистрирует последнюю обнаруженную ошибку.

2.3.1. ASSIGN. Подпрограмма ASSIGN связывает спецификацию файла с номером логического устройства. Обращение к подпрограмме ASSIGN должно предшествовать операторам ввода-вывода последовательного доступа READ и WRITE или оператору ввода-вывода DEFINE FILE, описывающего характеристики файла прямого доступа. Установленное соответствие между спецификацией файла и номером логического устройства сохраняется до закрытия файла с помощью подпрограммы CLOSE или оператора CLOSE или до конца выполнения программы.

Обращение: CALL ASSIGN (N, NAME, I, M, C, B)
где N — номер логического устройства (константа, переменная или выражение типа целый);
NAME — спецификация файла (устройство, имя и тип файла).
По умолчанию присваивается значение в соответствии с табл. 1;
I — характер обработки параметра NAME:
I>0, число обрабатываемых символов;
I=0, спецификация (NAME) обрабатывается до появления первого пробела или кода 000;
I<0, спецификация вводится пользователем с терминала после вывода *;
M — метод открытия файла (табл. 3);
C — управление выводом записей на терминал (табл. 4);
B — число буферов ввода-вывода. По умолчанию B=1.

В списке параметров обязательным является первый параметр, остальные могут быть опущены. Значения параметров в этом случае определяются по умолчанию. Если параметр включен в список параметров, то должны быть включены все предшествующие ему параметры.

Таблица 1

Номер устройства	Физическое устройство	Имя файла
1	2	3
1	Системное устройство (SY:)	FTN1.DAT
2	Устройство по умолчанию (резерв)	FTN2.DAT
3	Устройство по умолчанию (резерв)	FTN3.DAT
4	Устройство по умолчанию (резерв)	FTN4.DAT
5	Клавиатура терминала (TT:)	FTN5.DAT
6	Построчно-печатающее устройство (LP:)	FTN6.DAT
7	Печать терминала (TT:)	FTN7.DAT
8	Перфоленточное устройство ввода (PC:)	FTN8.DAT
9	Перфоленточное устройство вывода (PC:)	FTN9.DAT

Спецификация файла (параметр «NAME») может быть указана с переключателем (табл. 2). Если в спецификации указано только устройство и не указано имя файла, то оно рассматривается как устройство нефайловой структуры и должно использоваться для обмена данными файлов последовательного доступа. Применение такой спецификации к устройствам файловой структуры может привести к порче справочника этих устройств.

Таблица 2

ПЕРЕКЛЮЧАТЕЛИ ПАРАМЕТРА «NAME»

Переключатель	Назначение
1	2
/N	Запрещает режим управления возвратом каретки. По этому переключателю игнорируется значение параметра «C»
/C	Устанавливает режим управления возвратом каретки По этому переключателю игнорируется значение параметра «C»
/B:M	Определяет число буферов M для ввода — вывода. Параметр может принимать два значения: 1 или 2. По этому переключателю игнорируется значение параметра «B»

Таблица 3

ЗНАЧЕНИЕ ПАРАМЕТРА «M»

Значение параметра	Назначение
1	2
RDO	Файл только считывается. Если делается попытка записи в этот файл, возникает неустранимая ошибка. Если файл не найден, выводится на терминал сообщение об ошибке с номером 28

1	2
NEW	Создается новый файл с именем, указанным в спецификации, файл становится постоянным после закрытия его по подпрограмме CLOSE или после окончания работы программы. Выход из подпрограммы с помощью СУ/С не сохраняет файл
OLD	Файл определяется как существующий. Если он не найден, выводится на терминал сообщение об ошибке с номером 28
SCR	Файл определяется как временный. После закрытия файла или после выхода из программы он не сохраняется

По умолчанию значение параметра «M» определяется первым оператором ввода-вывода. Если первый оператор WRITE, то «M» принимает значение 'NEW', если — READ, то «M» принимает значение 'OLD'. Значения параметра «M» приведены в табл. 3.

Таблица 4

ЗНАЧЕНИЕ ПАРАМЕТРА C

Значение параметра	Назначение
1	2
NC	Выводятся на терминал все символы, включая первый. Запись ограничена вначале символом <ПС>, в конце — <ВК>
CC	Первый символ записи трактуется как символ управления кареткой и не выводится на терминал

По умолчанию параметр «C» принимает значение 'CC' для терминала, построчно-печатающего устройства и 'NC' для других устройств (табл. 4).

2.3.2. CLOSE. Подпрограмма CLOSE используется для закрытия файла на указанном логическом устройстве. Если файл открыт для записи, то перед его закрытием все частично заполненные буферы записываются в файл. Эти буферы освобождаются для дальнейшего использования и вся информация, предназначенная для записи в файл, игнорируется до тех пор, пока не будет повторного обращения к подпрограмме ASSIGN. Номер логического устройства становится свободным. В случае окончания программы пользователя по оператору STOP или при возникновении неустранимой ошибки, а также при обращении к подпрограмме EXIT происходит обращение к подпрограмме CLOSE и все открытые файлы закрываются.

Обращение: CALL CLOSE(N)
где N — номер логического устройства (константа, переменная или выражение типа целый).

2.3.3. DATE. Подпрограмма DATE используется для определения текущей даты (число, месяц, год), установленной в системе ФОДОС-2, в виде последовательности символов:
чч—ммм—гг
где чч — две цифры числа; ммм — три буквы месяца;
гг — последние две цифры года.
Обращение: CALL DATE(A)
где A — имя массива для записи последовательности символов (9 байт).

Параметром может быть имя вещественного массива, в котором используются первые три элемента, например: CALL DATE(A). В массиве A в первых трех элементах будет храниться текущая дата.

Параметром может быть элемент массива, например: CALL DATE(A(I)). В этом случае текущая дата хранится в элементах массива A(I), A(I+1), A(I+2).

2.3.4. IDATE. Подпрограмма IDATE используется для определения текущей даты (месяц, число, год), установленной в системе ФОДОС-2, в виде трех целых чисел.
Обращение: CALL IDATE(I, J, K)

Если текущая дата 16 марта 1987, то значения переменных будут следующими: I=3, J=16, K=87.
I=0, если дата в системе не установлена.

2.3.5. EXIT. Подпрограмма EXIT вызывает окончание работы программы пользователя: закрывает все открытые файлы и передает управление монитору. Подпрограмма EXIT эквивалентна выполнению оператора STOP за исключением того, что после выполнения подпрограммы отсутствует вывод на терминал.
Обращение: CALL EXIT

2.3.6. USEREX. Подпрограмма USEREX используется для вызова подпрограммы завершения, то есть подпрограммы, которой передается управление для окончания работы программы пользователя.
Обращение: CALL USEREX (NAME)
где NAME — имя подпрограммы завершения. 'Имя подпрограммы должно быть объявлено как внешнее имя.

После того как в основной программе будет подготовлено

все для окончания программы, управление передается программе завершения по команде безусловного перехода. Передача управления производится вместо обычного возврата управления монитору. Если пользователю необходимо передать управление в монитор, то это должна осуществлять программа завершения.

2.3.7. RANDU. Подпрограмма RANDU используется для генерации случайного числа.

Обращение: CALL RANDU (I(1), I(2), X)

где I(1) и I(2) — имена переменных типа целый;

X — переменная типа вещественный, принимающая значение случайного числа на интервале от 0 до 1.

Первоначально и для повторения последовательности случайных чисел необходимо I(1), I(2) установить в ноль.

Параметры I(1), I(2) могут принимать или нулевое значение, или то, которое получено на предыдущем шаге.

Использование подпрограммы RANDU подобно использованию основной внешней функции RAN.

2.3.8. SETERR. Подпрограмма SETERR используется для изменения реакции библиотеки ФОРТРАНа в случае обнаружения ошибки в программе пользователя.

Рекомендуется изменять реакцию библиотеки на ошибки с номерами от 1-го до 16-ти. В случае изменения реакции библиотеки на ошибки с номерами 0 и 20—68 может возникнуть неопределенная ситуация.

Обращение: CALL SETERR (N, M)

где N — номер ошибки библиотеки ФОРТРАНа (константа, имя переменной или выражение типа целый);

M — имя переменной или выражение типа целый, которое может принимать значения согласно табл. 5.

Таблица 5

Значение	Реакция по ошибке
1	2
0	Сообщение об ошибке выводится на терминал. Выполнение программы пользователя продолжается
1	Сообщение об ошибке выводится на терминал. Выполнение программы пользователя прекращается
2—127	Сообщение об ошибке выводится на терминал M — 1 раз Выполнение программы пользователя продолжается до M-го появления этой ошибки
128—255	Ошибка фиксируется, но сообщение об ошибке не выводится на терминал. Выполнение программы пользователя продолжается

2.3.9. ERRTST. Подпрограмма ERRTST используется для проверки возникновения указанной ошибки во время выполнения программы пользователя.

Обращение: CALL ERRTST (I, J)

где I — номер ошибки, для которой должна быть выполнена проверка;

J — имя переменной типа INTEGER*2 или имя элемента массива для регистрации ошибки:

J = 1 — есть ошибка; J = 2 — нет ошибки.

2.3.10. ERRSNS. Подпрограмма ERRSNS используется для регистрации последней возникшей ошибки.

Обращение: CALL ERRSNS (A, B)

где A — имя переменной или имя элемента массива типа целый для записи номера последней возникшей ошибки. Принимает нулевое значение, если ошибок не было;

B — имя переменной или имя элемента массива типа целый для записи номера логического устройства, если ошибка возникла во время операции ввода-вывода.

Подпрограмма ERRSNS может вызываться без параметров. В этом случае предыдущие данные об ошибке стираются, позволяя таким образом проверку ошибок в определенных программных секциях.

2.4. Программные секции

Объектные модули библиотеки ФОРТРАНа содержат пять программных секций (табл. 6).

Таблица 6

Имя секции	Признаки	Содержимое
1	2	3
OTS$I	RW, I, LCL, REL, CON	Коды команд и данные для модуля
OTS$P	RW, D, GBL, REL, OVR	Таблицы адресов других модулей библиотеки ФОРТРАНа
OTS$D	RW, D, LCL, REL, CON	Данные, к которым обращается модуль
OTS$S	RW, D, LCL, REL, CON	Дополнительная память, к которой обращается модуль
OTS$O	RW, I, LCL, REL, CON	Подпрограммы библиотеки ФОРТРАНа, которые могут размещаться или нет в области свопинга USR

2.5. Трассировка ошибок

Система ФОРТРАН для облегчения поиска ошибок в программе осуществляет трассировку ошибок. Трассировка оши-

бок позволяет определять место расположения ошибок во время выполнения программы.

Если ошибка произошла в подпрограмме SUBROUTINE или FUNCTION, то выводится на терминал имя подпрограммы и номер строки, в которой произошла ошибка, затем последовательно печатаются имена подпрограмм и номера тех строк, из которых происходило обращение к данной подпрограмме. Это позволяет точно определить место расположения ошибки.

Например, в результате выполнения программы

```
0001      A=0.0
0002      CALL SUB1(A)
0003      CALL EXIT
0004      END
0001      SUBROUTINE SUB1(B)
0002      CALL SUB2(B)
0003      RETURN
0004      END
0001      SUBROUTINE SUB2(C)
0002      CALL SUB3(C)
0003      RETURN
0004      END
0001      SUBROUTINE SUB3(D)
0002      E=1.0
0003      F=E/D
0004      RETURN
0005      END
```

на терминал выводится следующее сообщение:

```
?ERR 12 FLOATING  ZERO  DIVIDE
IN      ROUTINE   «SUB3»   LINE 3
FROM ROUTINE      «SUB2»   LINE 2
FROM ROUTINE      «SUB1»   LINE 2
FROM ROUTINE      «.MAIN.» LINE 2
```

Если в командной строке при трансляции программного модуля используется переключатель /NOLINENUMBERS (или /S), в тексте сообщения печатается вопрос вместо номера внутренней последовательности.

3. ОБРАЩЕНИЕ К ПРОГРАММЕ

Обращение к подпрограммам библиотеки ФОРТРАНа осуществляется из программы пользователя.

Обращение к внешним функциям языка ФОРТРАН

осуществляется путем указания в операторе имени функции.
Например, R=ABS(X)
где ABS — символическое имя функции.

Обращение к подпрограммам библиотеки ФОРТРАНа (см. п. 2.3) осуществляется с помощью оператора CALL:
CALL S(A1, A2, . . . , AN) или CALL S
где S — имя подпрограммы;
A1, A2, . . . , AN — фактические параметры.

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Входными и выходными данными для библиотеки ФОРТРАНа являются данные типа целый, вещественный, двойной точности, комплексный и текстовый. Внутреннее представление этих данных приведено в документе [2].

5. СООБЩЕНИЯ

Сообщения об ошибках выводятся на терминал во время выполнения программы в одной из двух форм: полной или краткой.
Полная форма сообщения: ?ERR NN текст
краткая форма сообщения: ?ERR NN
где NN — номер ошибки;
текст — краткое описание ошибки.

По умолчанию выдается полная форма сообщения. Сообщение в краткой форме выдается в том случае, если программа пользователя связана с модулем $SHORT библиотеки ФОРТРАНа (по переключателю /I в командной строке редактора связей). При использовании модуля $SHORT библиотеки ФОРТРАНа сохраняется 850 слов памяти. В файле на диске в формате .SAV или .REL из-за этого сохраняется 3—4 блока.

Ошибки, обнаруживаемые библиотекой ФОРТРАНа, подразделяются на четыре вида:
IG — ошибка фиксируется, но сообщение об ошибке не выводится на терминал. Выполнение программы пользователя продолжается;
W — сообщение об ошибке выводится на терминал, выполнение программы пользователя продолжается;
F — сообщение об ошибке выводится на терминал. Выполнение программы пользователя прекращается и управление передается монитору;
C:N — сообщение об ошибке выводится на терминал. Выполнение программы пользователя продолжается до N-го появ-

ления этой ошибки. В последнем случае эта ошибка считается неустранимой (F).

По подпрограмме SETERR библиотеки ФОРТРАНа можно изменить реакцию библиотеки на ошибки с номерами 1—16. Реакцию библиотеки на ошибки с номерами 0 и 20—68 изменять не рекомендуется во избежании непредвиденных ситуаций.

Если выполнение программы прервано вследствие неустранимой ошибки, рекомендуется использовать команду монитора CLOSE для закрытия открытых файлов, при этом часть информации, которая должна быть записана в файл, может быть потеряна.

Ниже приведены сообщения об ошибках с указанием вида ошибки и кратким пояснением.

0 NON—FORTRAN ERROR CALL
F

Причина. Эта ошибка может возникнуть в следующих случаях:
1) для основного задания с использованием подпрограммы завершения, которая вызывается подпрограммой системной библиотеки (SYSLIB), не было выделено достаточно места по команде FRUN/N:M для вызова программы завершения;
2) недостаточно памяти для фонового задания;
3) при работе с монитором SJ одна подпрограмма завершения прервала другую подпрограмму завершения;
4) при выполнении подпрограммы на языке АССЕМБЛЕР, связанной с программой на ФОРТРАНе, возникло прерывание по TRAP с неопознанным кодом ошибки.

Действие. 1) Определить объем памяти, требуемый для основного задания и подпрограммы завершения;
2) увеличить размер свободной области памяти на томе: стереть или переписать на другой том файлы, в которых нет необходимости, и сжать том;
3) использовать монитор FB для разрешения одновременного выполнения более одной подпрограммы завершения;
4) проанализировать программу на языке АССЕМБЛЕР.

1 INTEGER OVERFLOW
F

Причина. Результат операции умножения, деления или возведения в степень целого числа превысил значение 32767.

Действие. Исправьте программу. Используйте вещественные числа.

2 INTEGER ZERO DIVIDE
F

Причина. Деление на ноль при выполнении операции с целыми числами.

Действие. Исправьте программу.

3 COMPILER GENERATED ERROR
F

Причина. Исходный оператор не протранслирован полностью из-за ошибки.

Действие. Просмотрите листинг программы и исправьте ошибки, обнаруженные во время трансляции.

4 COMPUTED GOTO OUT OF RANGE
W

Причина. Значение переменной типа целый в операторе перехода по предписанию не совпадает со значением ни одной из меток списка или значение выражения в вычисляемом операторе перехода меньше единицы или больше числа меток списка.

Действие. Исправьте оператор.

5 INPUT CONVERSION ERROR
C:3

Причина. При вводе форматной записи обнаружен неверный символ.

Действие. Символу присваивается значение пусто. Исправьте неправильную запись.

6 OUTPUT CONVERSION ERROR
IG

Причина. При выводе форматной записи число символов в представлении числа превышает указанную ширину поля.

Действие. Поле вывода заполняется звездочками. Исправьте объявление формата.

10 FLOATING OVERFLOW
C:3

Причина. Результат операции превышает наибольшее допустимое вещественное число.

Действие. Результату присваивается нулевое значение. Исправьте программу.

11 FLOATING UNDERFLOW
IG

Причина. Результат операции меньше наименьшего допустимого вещественного числа.

Действие. Результату присваивается нулевое значение. Исправьте программу.

12 FLOATING ZERO DIVIDE
F

Причина. Деление на ноль при выполнении операции с вещественными числами.

Действие. Результату присваивается нулевое значение. Исправьте программу.

13 SQRT OF NEGATIVE NUMBER
C:3

Причина. Попытка вычислить квадратный корень из отрицательного числа.

Действие. Результату присваивается нулевое значение. Исправьте программу.

14 UNDEFINED EXPONENTIATION OPERATION
F

Причина. Попытка выполнить недействительную операцию возведения в степень.

Действие. Результату присваивается нулевое значение. Исправьте программу.

15 LOG OF ZERO OR NEGATIVE NUMBER
F

Причина. Попытка вычислить логарифм отрицательного числа или нуля.

Действие. Результату присваивается нулевое значение. Исправьте программу.

16 WRONG NUMBER OF ARGUMENTS
F

Причина. Основная внешняя функция из библиотеки ФОРТРАНа или подпрограмма из библиотеки системных подпрограмм вызвана с неправильным числом параметров.

Действие. Исправьте формат вызова функции или подпрограммы.

20 INVALID LOGICAL UNIT NUMBER
F

Причина. Номер логического устройства отличен от 1—99.

Действие. Исправьте оператор.

21 OUT OF AVAILABLE LOGICAL UNITS
F

Причина. Попытка одновременно открыть логических устройств больше допустимого числа (по умолчанию максимальное число — 6).

Действие. Вновь транслируйте исходную программу с переключателем /UNITS:M (N:M).

22 INPUT RECORD TOO LONG
F

Причина. Во время операции ввода размер записи превысил максимально допустимую длину (по умолчанию максимальная длина записи 136 байт).

Действие. Вновь транслируйте исходную программу с переключателем /RECORD:M (/R:M).

23 HARDWARE I/O ERROR
F

Причина. Аппаратная ошибка во время операции ввода-вывода.

Действие. Проверьте готовность и исправность оборудования. Проверьте диск на плохие блоки. Повторите операцию.

24 ATTEMPT TO READ/WRITE PAST END OF FILE
F

Причина. 1) Чтение после последней записи файла последовательного доступа;
2) ссылка на номер несуществующей записи файла прямого доступа;
3) на томе недостаточно места для записи файла.

Действие. 1) Используйте параметр «END=» в операторе ввода последовательного доступа;
2) исправьте программную логику в операторе ввода прямого доступа;
3) освободите место на томе или используйте другой том для операции записи.

25 ATTEMPT TO READ AFTER WRITE
F

Причина. Попытка считать выходной файл непосредственно после его записи на устройство последовательного доступа.

Действие. Исправьте программу. Перед считыванием файла с устройства последовательного доступа необходимо выполнить операцию REWIND или BACKSPACE.

26 RECURSIVE I/O NOT ALLOWED
F

Причина. Элементом списка вывода оператора WRITE является выражение, содержащее ссылку на внешнюю функцию, которая сама выполняет операцию ввода-вывода.

Действие. Исправьте программу.

27 ATTEMPT TO USE DEVICE NOT IN SYSTEM
F

Причина. Попытка использовать устройство, которое не определено в системе.

Действие. Присвойте с помощью команды монитора ASSIGN имя требуемого логического устройства или исправьте ошибку в операторе.

28 OPEN FAILED FOR FILE
F

Причина. 1) Заданный файл не найден;
2) на томе нет места;
3) выбранный канал уже использован.

Действие. 1) Проверьте наличие заданного файла;
2) освободите место на томе (сотрите ненужные файлы и сожмите том) или используйте другой том;
3) используйте другой канал.

29 NO ROOM FOR DEVICE HANDLER FODOS
F

Причина. Недостаточно места в памяти для драйвера заданного устройства.

Действие. Перепишите файл на системное устройство или устройство, драйвер которого является резидентным. Освободите часть оперативной памяти (удалите ненужные драйверы, снимите основное задание, используйте монитор SJ).

30 NO ROOM FOR BUFFERS
F

Причина. Недостаточно места в памяти для размещения буферов ввода-вывода.

Действие. 1) Уменьшите число логических устройств, которые одновременно открыты в момент обнаружения ошибки;
2) используйте один буфер вместо двух;
3) освободите часть оперативной памяти.

31 NO AVAILABLE I/O CHANNEL
F

Причина. Запрошено для ввода-вывода одновременно больше каналов, чем доступно в системе (15, исключая терминал).

Действие. Освободите каналы, которые не используются в данный момент.

32 FMTD—UNFMTD—RANDOM I/O TO SAME FILE
F

Причина. Комбинация форматного и бесформатного ввода-вывода для одного и того же файла.

Действие. Исправьте программу.

33 ATTEMPT TO READ PAST END OF RECORD
F

Причина. Попытка считать запись большей длины, чем фактически имеется в файле.

Действие. Проверьте длину записи в файле. Исправьте программу.

34 UNFMTD I/O TO TT OR LP
F

Причина. Попытка вывести бесформатную запись на терминал или печатающее устройство.

Действие. Назначьте другому логическому устройству, позволяющему вывод бесформатных записей, номер рассматриваемого устройства, используя команду монитора ASSIGN, подпрограмму ASSIGN библиотеки ФОРТРАНа или IASIGN библиотеки системных подпрограмм.

35 ATTEMPT TO OUTPUT TO READ ONLY FILE
F

Причина. Попытка записать файл, открытый для чтения.

Действие. Проверьте задание параметров при вызове подпрограммы ASSIGN библиотеки ФОРТРАНа или IASIGN библиотеки системных подпрограмм. Проверьте программу.

36 BAD FILE SPECIFICATION STRING
F

Причина. Не может быть обработан параметр в подпрограмме ASSIGN библиотеки ФОРТРАНа, указывающий на спецификацию файла.

Действие. Проверьте задание соответствующего параметра в подпрограмме ASSIGN.

37 RANDOM ACCESS READ/WRITE BEFORE DEFINE FILE
F

Причина. Попытка выполнить операцию чтения или записи прямого доступа до выполнения оператора DEFINE FILE.

Действие. Исправьте программу.

38 RANDOM I/O NOT ALLOWED ON TT OR LP
F

Причина. Попытка выполнить операцию ввода-вывода прямого доступа на терминале или печатающем устройстве.

Действие. Назначьте другому логическому устройству, позволяющему операции ввода-вывода прямого доступа, номер рассматриваемого устройства, используя команду монитора ASSIGN, подпрограмму ASSIGN библиотеки ФОРТРАНа или IASIGN библиотеки системных подпрограмм.

39 RECORD LARGER THAN RECORD SIZE IN DEFINE FILE
F

Причина. Длина записи больше указанной в операторе DEFINE FILE.

Действие. Уменьшите число элементов в списке ввода-вывода или переопределите длину записи.

40 REQUEST FOR A BLOCK LARGER THAN 65535
F

Причина. Попытка обратиться к блоку устройства, абсолютный адрес которого превышает 65535.

Действие. Исправьте программу.

41 DEFINE FILE ATTEMPTED ON AN OPEN UNIT
F

Причина. Попытка открыть файл с помощью оператора DEFINE FILE на устройстве, на котором уже открыт файл.

Действие. Закройте открытый файл, используя подпрограмму CLOSE библиотеки ФОРТРАНа, перед использованием DEFINE FILE.

42 MEMORY OVERFLOW COMPILING OBJECT TIME FORMAT
F

Причина. Переполнение памяти при задании спецификации формата в массиве.

Действие. Используйте объявление формата для операторов

ввода-вывода. Освободите часть оперативной памяти.

43 SYNTAX ERROR IN OBJECT TIME FORMAT
F

Причина. Синтаксическая ошибка в спецификации формата, заданной в массиве.

Действие. Исправьте спецификацию формата.

44 2ND RECORD REQUEST IN ENCODE/DECODE
F

Причина. Использование оператора ENCODE или DECODE более чем для одной записи

Действие. Исправьте объявление формата, связанное с ENCODE или DECODE (объявление может относиться только к одной записи), проверьте наличие символа «/» в объявлении формата.

45 INCOMPATIBLE VARIABLE AND FORMAT TYPES
F

Причина. Попытка вывести вещественные данные в формате целого числа или целые данные в формате вещественного числа.

Действие. Согласуйте описатели полей в объявлении формата с типом соответствующей переменной.

46 INFINITE FORMAT LOOP
F

Причина. Объявление формата не содержит описателей полей, используемых при передаче переменных в операторе ввода-вывода.

Действие. Исправьте объявление формата.

47 ATTEMPT TO STORE OUTSIDE PARTITION FODOS
F

Причина. Попытка записать элемент массива в ячейку, адрес которой находится за пределами памяти, отведенной для данного модуля.

Действие. Исправьте программу.

48 UNIT ALREADY OPEN
F

Причина. Попытка выполнить операцию, недопустимую для открытого файла.

Действие. Закройте файл, используя подпрограмму CLOSE библиотеки ФОРТРАНа или оператор CLOSE перед выполнением операции.

49 ENDFILE ON RANDOM FILE
F

Причина. Оператор ENDFILE содержит номер логического устройства, которое открыто для файла прямого доступа.

Действие. Исправьте программу.

50 KEYWORD VALUE ERROR IN OPEN STATEMENT
F

Причина. В операторе OPEN ключевому слову присвоено недопустимое значение.

Действие. Исправьте оператор OPEN.

51 INCONSISTENT OPEN/CLOSE STATEMENT SPECIFICATIONS
F

Причина. В операторе OPEN или CLOSE файл типа 'NEW' или 'SCRATCH' указан вместе с ключевым словом READONLY или ключевое слово READONLY используется вместе со значением ключевого слова DISP='DELETE'.

Действие. Исправьте оператор.

52 ATTEMPT TO DELETE A PROTECTED FILE
W

Причина. Указание ключевого слова DISP='DELETE' в операторе OPEN или CLOSE для защищенного файла.

Действие. Ликвидируйте защиту файла или исправьте оператор.

53 LIST—DIRECTED I/O SYNTAX ERROR
W

Причина. Синтаксическая ошибка в операторе ввода-вывода с преобразованием по списку.

Действие. Исправьте оператор.

59 USR NOT LOCKED FODOS
F

Причина. Не была подана команда монитора SET USR NOSWAP перед запуском программы в основном режиме, хотя при трансляции программы использовался переключатель /NOSWAP (/U).

Действие. Проанализируйте действие переключателя /NOSWAP и транслируйте программу либо без переключателя, либо команду SET USR NOSWAP перед запуском программы.

60 STACK OVERFLOWED
F

Причина. Это сообщение появляется при работе в фоновом режиме. Переполнение стека, что может привести к нарушению выхода из подпрограммы или нарушению выполнения операции открытия файла и трассировки.

Действие. Выделите дополнительную область для стека, используя переключатель /W:M при связывании объектных модулей. Проверьте программу.

61 ILLEGAL MEMORY REFERENCE
F

Причина. Обращение по несуществующему адресу памяти.

Действие. Если ошибка появилась в программе на языке АССЕМБЛЕРа, необходимо исправить исходную программу. Если ошибка появилась в программе на языке ФОРТРАН, следует убедиться, что библиотека ФОРТРАНа соответствует конфигурации технических средств.

62 FORTRAN START FAIL
F

Причина. Программа загружена в память, но не осталось свободной памяти для инициализации рабочей области и буферов библиотеки ФОРТРАНа.

Действие. При работе в фоновом режиме освободите часть оперативной памяти. При работе в основном задании используйте команду FRUN/N:M.

63 ILLEGAL INSTRUCTION
F

Причина. Попытка выполнить резервную операцию (например, операцию с плавающей точкой на ЭВМ, не имеющей аппаратных средств для работы с плавающей точкой).

Действие. Если ошибка появилась в программе на языке АССЕМБЛЕР необходимо исправить исходную программу. Если ошибка появилась в программе на языке ФОРТРАН, следует убедиться, что библиотека ФОРТРАНа соответствует конфигурации технических средств.

64 VIRTUAL ARRAY INITIALIZATION FAILURE
F

Причина. Транслятор не может инициализировать виртуальный массив. Ошибка может возникнуть в следующих случаях:

1) требования памяти для виртуальных массивов превышают имеющуюся оперативную память;
2) неправильно использована библиотека ФОРТРАНа. Программа может использовать виртуальные массивы при работе под управлением монитора SJ или FB при указании соответствующей библиотеки или при работе под управлением монитора XM без указания этой библиотеки;
3) отсутствуют технические средства для поддержки области логических адресов (PLAS) при работе под управлением монитора XM.

Действие. Уменьшите требования к объему виртуальной памяти за счет указания верхних границ с меньшими значениями. Правильно используйте библиотеку ФОРТРАНа.

65 VIRTUAL ARRAY MAPPING ERROR
F

Причина. Попытка обратиться за пределы расширенной памяти, отведенной для виртуальных массивов в программе.

Действие. Проверьте, чтобы значения индексов элементов виртуальных массивов, к которым происходит обращение, не превышали значения верхних границ.

66 UNSUPPORTED OPEN/CLOSE KEYWORD OR OPTION
F

Причина. В операторе OPEN или CLOSE указано ключевое слово или его значение, которое не поддерживается системой.

Действие. Исправьте оператор.

67 UNSUPPORTED OPEN/CLOSE KEYWORD OR OPTION
W

Причина. В операторе OPEN или CLOSE указано ключевое слово или его значение, которое не имеет смысла в данной системе.

Действие. Исправьте оператор.

68 DIRECT ACCESS RECORD SIZE ERROR
F

Причина. Размер записи в файле прямого доступа превышает 32767 двойных слов.

Действие. Исправьте программу.

ПРИЛОЖЕНИЕ

ПРОГРАММНЫЕ СЕКЦИИ

Порядок, в котором размещаются программные секции в программе, определяется порядком, в котором они обрабатываются редактором связей.

Головной модуль (обычно это первый модуль в последовательности входных файлов, указанных редактору связей) объявляет секции .PSECT в порядке, указанном в таблице.

Имя секции	Признаки
OTS$I	RW, I, LCL, REL, CON
OTS$P	RW, D, GBL, REL, OVR
SYS$I	RW, I, LCL, REL, CON
USER$I	RW, I, LCL, REL, CON
$CODE	RW, I, LCL, REL, CON
OTS$O	RW, I, LCL, REL, CON
SYS$O	RW, I, LCL, REL, CON
$DATAP	RW, D, LCL, REL, CON
OTS$D	RW, D, LCL, REL, CON
OTS$S	RW, D, LCL, REL, CON
SYS$S	RW, D, LCL, REL, CON
$DATA	RW, D, LCL, REL, CON
USER$D	RW, D, LCL, REL, CON
.$ $ $ $.	RW, D, GBL, REL, OVR
Другие блоки COMMON	RW, D, GBL, REL, OVR

Секции, содержащие команды, т. е. OTSI, OTSF, SYS$I, USER$I, $CODE и данные только для чтения (см. табл.), располагаются перед секциями, содержащими данные. Свопинг USR может выполняться только на секции, содержащие команды.

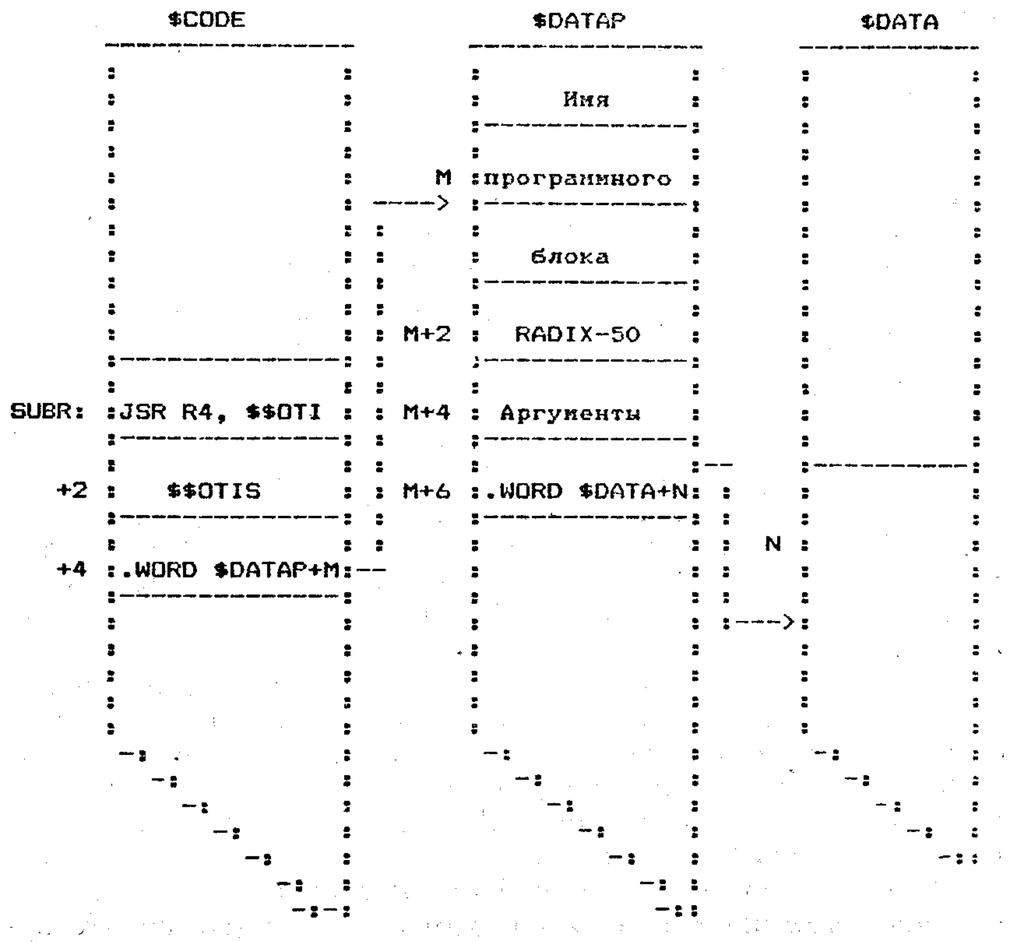
Подпрограммы написанные на языке АССЕМБЛЕР и вызывающие USR должны использовать те же программные секции, что и вызывающая их программа, написанная на языке ФОРТРАН. Команды и данные только для чтения необходимо поместить в секции USER$I, а все остальные данные — в секцию USER$D.

В программах с оверлейной структурой порядок следования программных секций в каждом оверлейном сегменте бу-

дет тот же, то есть корневой сегмент будет содержать секции только из команд и данных, которые не изменяются во время выполнения программы.

Для отладки программы пользователя с помощью программы отладки необходимо иметь листинг программы на языке АССЕМБЛЕР и карту загрузки.

В карте загрузки приводится базовый адрес программной секции $CODE. Базовый адрес нужного (отыскиваемого) блока программы соответствует точке входа данного блока программы. Адрес секции $DATAP, соответствующий данному блоку программы, содержится по адресу: базовый адрес блока программы +4. Базовый адрес секции $DATA содержится по адресу: базовый адрес секции $DATAP+6. Связь между адресами секций $CODE, $DATAP, $DATA приведена на нижеследующем рисунке.



ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Операционная система ФОДОС-2
   Редактор связей
   Руководство оператора
2. ФОРТРАН/ФОДОС-2
   Транслятор с ФОРТРАНа
   Руководство программиста

ДИАЛОГОВЫЙ ОТЛАДЧИК. РУКОВОДСТВО ПРОГРАММИСТА

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

1.1. Назначение

Программа «Диалоговый отладчик» (файл FDT.OBJ) предназначена для отладки программ, написанных на языке ФОРТРАН.

Программа «Диалоговый отладчик» позволяет выявлять ошибки в отлаживаемой программе путем прогона программы пользователя определенными участками и проверки на ожидаемые результаты в различных точках. Во время отладки программы можно изменять исходные данные.

Для исправления ошибок, обнаруженных во время отладки, необходимо внести соответствующие изменения в исходный модуль программы и заново его протранслировать.

1.2. Условия применения

1.2.1. Требования к техническим средствам

Для работы программы «Диалоговый отладчик» необходима следующая минимальная конфигурация технических средств:

— центральный процессор;
— оперативное запоминающее устройство емкостью 16К слов для работы под управлением мониторов SJ или FB, 32К слов и диспетчер памяти для работы под управлением монитора XM;
— устройство ввода-вывода информации на гибких магнитных дисках или на жестком магнитном диске;
— таймер;
— терминал.

1.2.2. Требования к программным средствам

1) Для работы с программой «Диалоговый отладчик» необходимы следующие программные средства:
— компоненты операционной системы ФОДОС-2:

FMON<YY>.SYS — монитор системы;
SWAP.SYS — файл свопинга;
<XX>.SYS — драйвер дисков;
LINK.SAV — редактор связей;
SYSLIB.OBJ — библиотека системных подпрограмм
где YY — SJ, FB, XM
XX — DX, DY, DW, MX, MY или DXX,
DYX, DWX, MXX, MYX
— компоненты операционной системы ФОРТРАН/ФОДОС-2:
FORLIB.OBJ — библиотека ФОРТРАНа.

2) При работе с программой «Диалоговый отладчик» необходимо иметь:
объектные модули отлаживаемой программы в нанизанном коде (THR) с номерами внутренней последовательности;
— листинг отлаживаемой программы, содержащий поле исходной программы и поле карты памяти.

Описание получения объектного модуля и листинга программы приведено в документе [2].

2. СПОСОБ ОПИСАНИЯ ПРОГРАММЫ

В документе используются следующие соглашения:
— элемент или группа элементов, заключенные в квадратные скобки ([]), необязательны;
— элемент или группа элементов, предшествующие многоточию (...), могут быть повторены;
— элементы из латинских букв, являющиеся ключевыми словами, записываются точно также, как они записаны в формате команды;
— ключевые слова в форматах команд и командные строки в примерах подчеркнуты.

3. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

Отладка программы на языке ФОРТРАН осуществляется с помощью команд диалогового отладчика в режиме диалога пользователя с ЭВМ.

Команды отладчика (см. табл. 1) позволяют управлять выполнением отлаживаемой программы, выводить на терминал значение любых переменных или элементов массивов, а также изменять значения переменных или элементов массивов на время прогона отлаживаемой программы.

Из команд отладчика можно сформировать командную строку — последовательность команд, разделенных между собой точкой с запятой.

Из команд отладчика можно также сформировать макроопределение, которое может быть вызвано неоднократно с помощью макрокоманды. Макроопределение может размещаться на нескольких строках.

Готовность отладчика принять командную строку или макроопределение определяется по выводу на терминал символа «!». Команда или командная строка будет выполняться после ввода символа <ВК>.

Команды, командные строки, макроопределения вводятся пользователем с терминала во время пауз. Первая пауза — автоматическая, осуществляется после запуска отладчика. Остальные паузы задаются по командам отладчика.

После завершения отладки или выполнения отлаживаемой программы, а также по команде STOP отладчика (п. 5.1.4) управление передается операционной системе.

3.1. Команды диалогового отладчика имеют следующий вид:

W [P1[,P2] ...]

где W — символическое имя команды (см. табл. 1);

каждое PI — константа или символическое имя, называемое параметром команды.

Таблица 1

Имя команды	Назначение
1	2
ACCEPT	Вводит новое значение переменной или элемента массива
CONTINUE	Продолжает выполнение программы
DIMENSION	Связывает спецификацию ячейки с описанием массива
ERASE	Уничтожает связь адреса ячейки с именем переменной или описанием массива
GOTO	Изменяет порядок выполнения команд в макроопределении
IF	Выполняет указанную команду, если отношение принимает значение «истина»
MACRO	Определяет, переопределяет, выполняет или уничтожает макроопределение
NAME	Связывает спецификацию ячейки с именем переменной
PAUSE	Устанавливает входную или операторную паузу
RESET	Отменяет входную или операторную паузу
START	Повторно запускает отлаживаемую программу
STEP	Устанавливает пошаговую паузу
STOP	Завершает процесс отладки, передает управление операционной системе
TYPE	Выводит на терминал значения переменных или элементов массива
WATCH	Устанавливает паузу слежения
WHAT	Выводит на терминал текущее состояние отладчика

Имя команды может быть сокращено до трех первых символов. Пробелы внутри имени команды недопустимы. Имени команды может предшествовать любое число пробелов.

Параметром может быть константа целая или текстовая, имя переменной, имя элемента массива, имя массива, описание массива, спецификация ячейки.

Параметры должны быть отделены друг от друга запятой и от имени команды — хотя бы одним пробелом.

3.1.1. Спецификация ячейки. Определяет адрес данного и его тип и используется для обращения к данным программы пользователя.

Спецификация ячейки имеет вид: Y[/T]

где Y — указатель адреса ячейки;

T — указатель типа.

Указателем адреса ячейки является смещение, именованный, относительный или индексированный адрес. При обращении к переменным указатель адреса задается в виде смещения, именованного или относительного адреса; при обращении к элементам массива — в виде индексированного адреса.

Указатель типа указывает на тип данного. Тип данного определяет внутреннее представление этого данного в памяти машины (см. [2]).

3.1.1.1. Смещение — это разность между адресом переменной и адресом начала блока данных данного программного модуля. Смещение задается в одном из видов:

S+X;

B+X — для переменных именованного общего блока;

.BCOM.+X — для переменных неименованного общего блока,

где S — имя программного модуля;

B — имя общего блока;

X — величина смещения.

Величина смещения определяется по листингу программы. Она указана в поле карты памяти в колонке OFFSET (см. [2]). При задании величины смещения достаточно указать значащие цифры.

Если головной модуль отлаживаемой программы не имеет имени, то отладчик присваивает ему имя .MAIN..

Имя текущего программного модуля можно не указывать. Текущим программным модулем является модуль, который выполняется в данный момент времени. Это может быть головной модуль, модуль — подпрограмма, модуль — функция.

Имя общего блока должно быть определено по команде NAME (п. 5.2.1.1).

3.1.1.2. Именованный адрес — это адрес, с которым связано имя переменной.

Связь адреса с именем переменной осуществляется по команде NAME (п. 5.2.1.1). В дальнейшем ссылка на именованный адрес производится при помощи имени. С адресом может быть связано только одно имя.

3.1.1.3. Относительный адрес — это адрес, который определяется с помощью именованного адреса в виде:

V+X

где V — именованный адрес;

X — разность между величиной смещения указываемой ячейки и величиной смещения именованного адреса.

Относительные адреса используются для доступа к данным, расположенным в последовательных ячейках памяти.

3.1.1.4. Индексированный адрес — это адрес, с которым связано описание массива.

Связь адреса с описанием массива осуществляется по команде DIMENSION (п. 5.2.1.2). Индексированный адрес указывает на первый элемент массива. Адрес любого элемента массива задается именем массива, за которым следует индекс, например: A(3,7).

3.1.1.5. Указатель типа. Указатели типов и типы данных приведены в табл. 2.

Для целых данных указатель типа I в спецификации ячейки можно не указывать, если указатель адреса задается в виде смещения.

Если в отлаживаемой программе объявление формата не содержит коды преобразований D, E, F и G, то соответствующие подпрограммы преобразования чисел с плавающей точкой не включаются редактором связей в загрузочный модуль отлаживаемой программы.

Диалоговый отладчик может работать с данными, указатель типа которых C, D, E, при условии использования в отлаживаемой программе кодов преобразования D, E, F или G или при включении библиотечного модуля ФОРТРАНа RCI$ в загрузочный модуль программы.

Примеры:

204/E

спецификация ячейки для переменной: смещение — 204, указатель типа — E.

16/PI

спецификация ячейки для переменной, используемой в качестве формального параметра: смещение — 16, указатель типа — I. P показывает, что данная переменная является формальным параметром.

.BCOM.+20

спецификация ячейки для переменной из неименованного общего блока: смещение — 20, указатель типа — I (по умолчанию).

FUN+30

спецификация ячейки для переменной модуля FUN: смещение — 30, указатель типа — I (по умолчанию).

A/J

спецификация ячейки для переменной: именованный адрес — A, указатель типа — J.

B+20/E

спецификация ячейки для переменной: относительный адрес — B+20, указатель типа — E.

D(5)/E

спецификация ячейки для элементов массива: индексированный адрес — D(5), указатель типа — E.

3.2. Паузы

Команды отладчика подаются во время приостановки выполнения отлаживаемой программы — паузы. Имеются пять видов пауз:

— автоматическая;
— входная;
— операторная;
— пошаговая;
— слежения.

Все паузы, кроме автоматической, устанавливаются по командам отладчика. Пауза может быть установлена на любой оператор отлаживаемой программы, имя любой подпрограммы или функции.

Автоматическая пауза осуществляется после запуска программы (п. 4.2).

Входная пауза осуществляется на входе в указанный модуль-подпрограмму или модуль-функцию отлаживаемой программы и устанавливается по команде PAUSE (п. 5.1.1.1).

Операторная пауза осуществляется перед выполнением

указанного оператора отлаживаемой программы и устанавливается по команде PAUSE (п. 5.1.1.1).

Пошаговая пауза осуществляется после выполнения указанного числа операторов отлаживаемой программы и устанавливается по команде STEP (п. 5.1.1.2).

Пауза слежения осуществляется при изменении значения указанной переменной или элемента массива и устанавливается по команде WATCH (п. 5.1.1.3).

Паузы не могут быть установлены на системные подпрограммы, основные внешние функции ФОРТРАНа, а также на входе в подпрограмму, написанную на языке АССЕМБЛЕР.

4. ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1. Формирование загрузочного модуля

Для работы с программой «Диалоговый отладчик» необходимо файл FDT.OBJ связать с отлаживаемой программой с помощью программы «Редактор связей» (см. [1]) и получить единый загрузочный модуль.

Формат командной строки для получения загрузочного модуля: R LINK

* выхспф[,карспф]=входспф, бибспф, отлспф

где выхспф — спецификация файла загрузочного модуля;
карспф — спецификация файла карты загрузки;
входспф — спецификация файла отлаживаемой программы;
бибспф — спецификация файла FORLIB.OBJ, спецификация файла SYSLIB.OBJ;
отлспф — спецификация файла FDT.OBJ.

Спецификация файла карты загрузки указывается в том случае, если программа пользователя содержит общие блоки данных.

Спецификация файла SYSLIB.OBJ указывается в том случае, если файл SYSLIB.OBJ находится не на системном носителе.

4.2. Вызов программы «Диалоговый отладчик» осуществляется по команде монитора RUN.

Формат командной строки для вызова программы:
RUN выхспф
где выхспф — спецификация файла загрузочного модуля (п. 4.1).

После вызова отладчик выводит на терминал номер своей версии: FDT B01.00 и устанавливает автоматическую паузу FDT PAUSE AT ISN M IN S

где M — номер внутренней последовательности первого оператора головного модуля;
S — имя головного модуля.

Если головной модуль не имеет имени, отладчик присваивает ему имя .MAIN..

После печати символа «!» отладчик ожидает ввода командной строки.

Если необходимо выполнить программу без вмешательства отладчика, достаточно подать команду START или CONTINUE (п. 5.1.2, 5.1.3) во время автоматической паузы.

5. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Входными данными для программы «Диалоговый отладчик» являются вводимые с терминала команды, командные строки, макроопределения, а выходными — сообщения о выполнении команд.

Существуют команды трех типов:
— команды управления отлаживаемой программой;
— команды организации ввода-вывода;
— команды управления отладчиком.

5.1. Команды управления отлаживаемой программой

К командам управления отлаживаемой программой относятся:
— команды пауз;
— команда повторного запуска;
— команда продолжения;
— команда останова.

Команды этой группы используются для установки и отмены пауз, повторного запуска отлаживаемой программы, прекращения или продолжения выполнения программы.

5.1.1. Команды пауз

5.1.1.1. Команда входной и операторной паузы

Команда входной паузы имеет вид:

PAU[SE] S [AFT[ER] N] [MAC[RO] L]

команда операторной паузы имеет вид:

PAU[SE] [S], M [AFT[ER] N] [MAC[RO] L]

где S — имя программного модуля;
M — порядковый номер внутренней последовательности программного модуля, на котором приостанавливается выполнение отлаживаемой программы;
N — счетчик проходов (целое число от 1 до 32767);

L — номер макрокоманды (целое число от 0 до 7).

Перед и после слов «AFTER», «MACRO» должно быть не менее одного пробела.

По команде PAUSE в отлаживаемой программе устанавливается входная или операторная пауза. При прохождении программы через установленную паузу происходит приостановка выполняемой программы и на терминал выводится сообщение:

FDT PAUSE AT ISN M IN S

где M — порядковый номер внутренней последовательности текущего программного модуля, на котором приостановилось выполнение отлаживаемой программы;
S — имя текущего программного модуля.

Параметр «AFTER N» указывает сколько раз управление должно быть передано на отмеченный паузой программный модуль или оператор, прежде чем произойдет пауза. По умолчанию значение N равно единице. Текущее значение N можно распечатать по команде WHAT (п. 5.3.3).

Параметр «MACRO L» определяет макрокоманду, которая должна быть выполнена прежде чем произойдет входная или операторная пауза. В этом случае во время паузы на терминал выводится только «!».

Если в команде PAUSE параметр «MACRO L» отсутствует, то будет выполняться макрокоманда с нулевым номером, если соответствующее макроопределение определено к этому времени.

Максимальное число входных и операторных пауз, которое одновременно может быть установлено по команде PAUSE, равно восьми. Если по команде PAUSE на один оператор отлаживаемой программы установлено две паузы, то предыдущее определение паузы отладчиком заменяется на последующее. Если в команде PAUSE указано несуществующее имя программного модуля или несуществующий порядковый номер внутренней последовательности программного модуля, то пауза не устанавливается и сообщение об ошибке не выдается.

Если к моменту выполнения команды PAUSE макроопределение, указанное в команде, не определено, то параметр «MACRO L» игнорируется.

Примеры:

. PAUSE,10 AFTER 6 MACRO 2

устанавливается пауза на оператор с порядковым номером внутренней последовательности текущего программного модуля 10, счетчик проходов равен 6-ти. Перед приостановкой

выполнения программы выполняется макрокоманда с номером 2.

: PAUSE MIN

устанавливается пауза на входе в модуль-функцию с именем MIN.

5.1.1.2. Команда пошаговой паузы имеет вид:

STE[P] [N]

где

N — целое число, указывающее на число выполняемых операторов отлаживаемой программы. По умолчанию N равно единице.

По команде STEP продолжается выполнение отлаживаемой программы от текущей установленной паузы, при этом установленные ранее операторные и входные паузы игнорируются. После выполнения N операторов отлаживаемой программы происходит приостановка выполняемой программы и на терминал выводится сообщение (см. п. 5.1.1.1).

Команды, следующие за командой STEP в командной строке, игнорируются.

5.1.1.3. Команда паузы слежения имеет вид:

WAT[CH] C

где

C — спецификация ячейки.

В спецификации ячейки указатель типа может быть любым, кроме R, AN, Z.

По команде WATCH в отлаживаемой программе устанавливается пауза слежения. При прохождении программы через ячейку с указанным адресом происходит приостановка отлаживаемой программы, если содержимое ячейки изменяется, при этом на терминал выводится сообщение:

WATCH PAUSE
FDT PAUSE AT INS M IN S

где M — порядковый номер внутренней последовательности программного модуля, на котором произошла приостановка отлаживаемой программы;
S — имя текущего программного модуля.

Задание другой команды WATCH отменяет ранее установленную паузу и устанавливает новую паузу слежения.

По команде WATCH без параметра отменяется ранее установленная пауза слежения.

Пауза слежения не зависит от установки других пауз. Можно установить паузу слежения одновременно с другой

паузой на любой оператор отлаживаемой программы. При этом первой выполняется операторная или пошаговая пауза, так как прежде чем произойдет пауза слежения, должен выполниться оператор, изменяющий содержимое ячейки по указанному адресу. Это возможно только после продолжения выполнения программы.

Примеры:

```
! WATCH V
```

при изменении значения переменной V произойдет пауза слежения.

```
! WATCH 205/E
```

при изменении значения данного, адрес которого задан смещением 205, произойдет пауза слежения.

5.1.1.4. Команда отмены входной и операторной паузы

Команда отмены входной паузы имеет вид:

```
RES[ET] S,
```

команда отмены операторной паузы имеет вид

```
RES[ET] [S],M
```

где

S — имя программного модуля;

M — порядковый номер внутренней последовательности программного модуля.

По команде RESET отменяется входная или операторная пауза.

Примеры:

```
! RESET SUBR
```

отменяется входная пауза, установленная на модуль-подпрограмму с именем SUBR.

```
! RESET ,10
```

отменяется операторная пауза, установленная на оператор с порядковым номером внутренней последовательности 10.

5.1.2. Команда повторного запуска имеет вид:

```
STA[RT]
```

по команде START осуществляется повторный запуск отлаживаемой программы. Команда не отменяет и не изменяет ранее заданные определения имен и пауз. Команда START игнорируется, если в момент ее подачи в отлаживаемой программе есть открытые файлы; на терминал в этом случае выводится символ «!».

Команды, следующие за командой START в командной строке, игнорируются.

5.1.3. Команда продолжения имеет вид:

```
CON[TINUE] [N]
```

где

N — целое число, называемое счетчиком проходов. По умолчанию N равно единице.

По команде CONTINUE продолжается выполнение отлаживаемой программы от текущей паузы.

Параметр N используется при продолжении программы от паузы, установленной по команде PAUSE, и указывает, сколько раз должна пройти программа через эту паузу, прежде чем произойдет эта же пауза. В остальных случаях (для пошаговой паузы и паузы слежения) параметр N в команде CONTINUE игнорируется. Команду CONTINUE с параметром обычно используют внутри цикла отлаживаемой программы.

Команды, следующие за командой CONTINUE в командной строке, игнорируются.

5.1.4. Команда останова имеет вид:

```
STO[P]
```

по команде STOP завершается процесс отладки: все открытые файлы закрываются и управление передается операционной системе.

Команды, следующие за командой STOP в командной строке, игнорируются.

5.2. Команды организации ввода-вывода

Командами организации ввода-вывода являются:

— команды связи;

— команда ввода;

— команда вывода.

Команды этой группы используются для связи спецификации ячейки с именем переменной или описанием массива, изменения значений переменных или элементов массивов на время прогона отлаживаемой программы, а также вывода на терминал текущих значений переменных или элементов массива.

5.2.1. Команды связи

5.2.1.1. Команда связи с именем переменной имеет вид:

```
NAM[E] V, C
```

где

V — имя переменной;

C — спецификация ячейки.

Указателем адреса ячейки не может быть индексированный адрес.

Имя переменной содержит не более шести символов. Первый символ должен быть буквой, остальные — либо буквой, либо цифрой.

По команде NAME связывается спецификация ячейки с именем переменной.

Если имя переменной, указанное в команде NAME, определено ранее, то предыдущее определение имени переменной заменяется последующим. Если в команде NAME отсутствует спецификация ячейки, то уничтожается связь имени с адресом ячейки, т. е. адрес перестает быть именованным.

По команде NAME адрес общего блока данных программы пользователя связывается с именем B:

NAME B,.ABS.+A

где

B — имя;

A — абсолютный адрес общего блока (определяется по карте загрузки отлаживаемой программы).

Примеры:

```
! NAME I, 202/J
```

ячейка, заданная смещением 202 и указателем типа J, связывается с именем переменной I.

```
! NAM DELTA
```

предыдущее определение имени DELTA уничтожается.

5.2.1.2. Команда связи с описанием массива имеет вид:

DIM[ENSION]V (I1[, I2]...), C

где

V(I1 [, I2] ...) — описание массива;

каждое IJ — размер по измерению;

C — спецификация ячейки.

По команде DIMENSION связывается спецификация ячейки с описанием массива. Если в команде DIMENSION отсутствует спецификация ячейки, то предыдущая связь этого массива с адресом уничтожается.

Примеры:

```
! DIMENSION A(10, 5), 46/E
```

ячейка со смещением 46 и указателем типа E связывается с описанием массива A(10, 5).

```
! DIM W(6)
```


описание массива W(6) теряет свою связь с адресом ячейки.

5.2.1.3. Команда отмены связи имеет вид:

ERA[SE] V1[, V2] ...

где

каждое VI — имя переменной или имя массива.

По команде ERASE уничтожаются связи адресов ячеек с именами переменных и описаниями массивов.

Пример.

```
! ERASE D, L
```

переменные D и L теряют свою связь с адресами ячеек.

5.2.2. Команда ввода имеет вид:

ACC[EPT] Q1[Q2] ...

где каждое QI — спецификация ячейки, буквенно-цифровой литерал или оператор прямого присваивания.

Оператор прямого присваивания записывается в виде:

C=E

где C — спецификация ячейки;

E — константа любого типа, именованный или индексированный адрес.

По команде ACCEPT присваивается переменной или элементу массива новое значение. После вывода на терминал символа «?» пользователь должен ввести с терминала новое значение переменной или элемента массива. Формат ввода данных приведен в табл. 2.

Логические константы и константы RADIX-50 должны вводиться с терминала с предшествующим константе символом «$». Логические константы задаются как «T» или «F».

Комплексные константы вводятся как две вещественные константы, разделенные запятой (действительная и мнимая часть соответственно). Числовые константы могут содержать до 40 цифр.

Таблица 2

Указатель типа	Тип данных	Длина в байтах	Формат ввода-вывода данных
1	2	3	4
I	INTEGER*2	2	Целое число
J	INTEGER*4	4	Целое число
L	LOGICAL*4	4	Логическое данное
M	LOGICAL*1	1	Логическое данное

1	2	3	4
E	REAL*4	4	Вещественное данное
D	REAL*8	8	Данное двойной точности
C	COMPLEX	8	Комплексное данное
B	BYTE	1	Десятичная цифра
R	BYTE	2	Текстовая константа RADIX—50
O	BYTE	2	Восьмеричное данное
AN	BYTE	N	Текстовое данное
Z	BYTE	N	Текстовое данное

Указатель типа Z используется в подпрограмме, работающей с текстовыми данными.

Пример.

```
! ACCEPT I=5, 'K=', K
K=?      10
```

Переменной I присваивается значение 5, выводится буквенно-цифровой литерал "K=", символ «?», по которому запрашивается значение для переменной K, и присваивается введенное значение 10 переменной K.

Примеры:

```
ACCEPT 202=1
```

в ячейку со смещением 202 записывается значение 1.

```
ACCEPT DELTA=EPSI
```

содержимое именованной ячейки DELTA заменяется содержимым именованной ячейки EPSI.

```
ACCEPT I=0, J=1, 256/E=2.71828
```

В именованные ячейки I и J записываются ноль и единица соответственно, в ячейку со смещением 256 записывается значение 2.71828.

5.2.3. Команда вывода имеет вид:

```
TYP[E] Q1[, Q2] ...
```

где каждое QI — спецификация ячейки, буквенно-цифровой литерал, именованный или относительный адрес.

По команде TYPE выводятся на терминал значения указанных переменных и элементов массива отлаживаемой программы. Формат вывода данных приведен в табл. 2.

Индексы в элементе массива задаются либо целыми константами, либо ранее определенными по команде NAME именами типа целый. Число индексов не должно превышать числа измерений, указанных в команде DIMENSION. Если для

элемента массива будет указано меньше индексов, чем в описании массива, то отсутствующие правые индексы полагаются равными единице. Например, для трехмерного массива D указание D(5) воспринимается как D(5, 1, 1).

Пример:

```
! TYPE 'DELTA', A, K(1, 3)
DELTA, 52.3, 68
```

По команде TYPE выводится на терминал буквенно-цифровой литерал DELTA, значения переменной A и элемента массива K(1, 3), равные 52.3 и 68 соответственно.

5.3. Команды управления отладчиком. Командами управления отладчиком являются:

— команды макросредств;
— команда условного перехода;
— команда печати текущего состояния отладчика.

Команды этой группы используются для определения, выполнения и уничтожения макроопределения, изменения порядка выполнения команд и вывода текущего состояния отладчика.

5.3.1. Команды макросредств

5.3.1.1. Команда MACRO. По команде MACRO можно определить, переопределить или уничтожить макроопределение и выполнить макрокоманду. Команда макроопределения имеет вид:

```
MACRO L([K1]F1[; K2]F2] ...)
```

Макрокоманда имеет вид:

```
MACRO L
```

Команда уничтожения макроопределения имеет вид:

```
MACRO L( )
```

где L — номер макроопределения (целое число от 0 до 7);
[K1]F1[;[K2]F2] ... — тело макроопределения;
каждое FI — команда диалогового отладчика;
каждое KI — метка команды диалогового отладчика (целое число от 1 до 32767);
( ) — ограничители, парные круглые скобки.

Пробелы между номером макроопределения и левой скобкой недопустимы. После метки должен быть хотя бы один пробел. Макроопределение переопределяется вводом нового тела макроопределения.

Макроопределения вводятся во время пауз. Тело макроопределения может быть расположено на нескольких строках. Готовность отладчика принять очередную строку тела

макроопределения определяется выводом на терминал восклицательного знака.

По команде MACRO L вызывается макроопределение с номером L. Макроопределение прекращает свое выполнение при обнаружении конца макроопределения — правой скобки, а также и при появлении команд STOP, CONTINUE и START в теле выполняемого макроопределения.

По команде MACRO L( ) макроопределение с номером L уничтожается. Правая и левая скобки не должны разделяться какими-либо символами, включая пробел.

Пример.

```
! MACRO 5(100 TYPE Y  ;IF C<12;
! GOTO 100; TYPE A(2))
! MACRO 5
16
3.55
```

По команде MACRO вызывается макроопределение с номером 5 и выполняется тело макроопределения: на терминал выводится значение переменной Y, равное 16-ти и значение элемента массива A(2), равное 3.55. Отладчик ожидает ввода с терминала новой командной строки. Макроопределение с номером 5 расположено на двух строках.

5.3.1.2. Команда безусловного перехода имеет вид:

```
GOTO K
```

где

K — метка команды отладчика в макроопределении, на которую передается управление (целое число от 1 до 32767).

По команде GOTO изменяется порядок выполнения команд отладчика при выполнении макрокоманды. Если в макроопределении используется несколько одинаковых меток, то отладчик передает управление на первую.

Команды диалогового отладчика GOTO и IF могут быть использованы для задания цикла в макроопределении. Для выхода из бесконечного цикла нужно подать с терминала команду СУ/С дважды.

Для перезагрузки отлаживаемой программы необходимо использовать команду монитора RUN.

5.3.2. Команда условного перехода имеет вид:

```
IF C@E ; W
```

где

C — спецификация ячейки;

@ — знак операции отношения;
E — спецификация ячейки, константа;
W — команда отладчика.

По команде IF сначала вычисляется отношение C@E. Если отношение принимает значение «истина», то выполняется команда W, в противном случае управление передается на следующую команду отладчика.

Знаки операции отношения приведены в табл. 3.

Таблица 3

Знак операции	Представляемая операция
1	2
=	Равно
<>	Не равно
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно

Спецификации «C» и «E» должны иметь один и тот же указатель типа. Если указатели типа не совпадают, то отладчик производит сравнение по указателю типа, заданного в «C». Допустимыми указателями типа являются E, I, J.

Пример:

```
! IF  130/E <> 6.5 ; TYPE C; CONTINUE
```

Будет выполняться команда «TYPE C», если значение данного по адресу, определенному смещением 130, не равно 6.5, в противном случае, будет выполняться команда CONTINUE.

5.3.3. Команда печати текущего состояния отладчика имеет вид:

```
WHA[T]
```

По команде WHAT на терминал выводится текущее состояние отладчика (все установленные операторные и входные паузы и макроопределения):

```
PAUSES:
  S  M  [AFTER  N] [MACRO L]
  S  [AFTER N] [MACRO L]
        .
        .
        .
```

```
MACROS:
   L:  F1[;F2]...
       •
       .
       .
```

где S — имя программного модуля;
M — порядковый номер внутренней последовательности программного модуля, на котором установлена операторная пауза;
N — счетчик проходов;
L — номер макроопределения;
каждое FI — команда отладчика.

Пример:

```
! WHAT
  ____
PAUSES:
OTL 5 AFTER 2  MACRO 1
FUN 15
MACROS:
1:  TYPE A; TYPE C
3:  TYPE K; 2 TYP C; IF B(1)<11; GOTO 2; TYP B(2)
```

В данный момент отладчиком установлены:
— операторная пауза на операторе с порядковым номером внутренней последовательности 5-го головного модуля с именем OTL со счетчиком проходов 2 и макрокомандой 1;
— операторная пауза на операторе с порядковым номером внутренней последовательности 15-го модуля-функции с именем FUN;
— макроопределение с номером 1;
— макроопределение с номером 3.

6. СООБЩЕНИЯ

Сообщения об ошибках выводятся на терминал непосредственно после ввода команды или командной строки.

FDT START FAIL

Причина. Объектный модуль программы не содержит номеров внутренней последовательности.

Действие. Получите объектный модуль с номерами внутренней последовательности.

% SUBSCR OUT OF BOUNDS

Причина. Предупреждающее сообщение. Недопустимое значение индекса или значение индекса не определено.

Действие. Правильно укажите индекс при ссылке на элемент массива.

? BAD DIM

Причина. В команде DIMENSION использована неверная спецификация ячейки или недопустимые значения индексов в описании массива.

Действие. Исправьте команду DIMENSION.

? BAD LOC

Причина. В спецификации ячейки неверный указатель типа ячейки или недопустимый указатель адреса ячейки.

Действие. Исправьте спецификацию ячейки.

? BAD MACRO

Причина. В теле макроопределения есть макроопределение с тем же номером, что данное макроопределение. Сообщение выводится во время выполнения макрокоманды.

Действие. Уничтожьте данное макроопределение или правильно его определите.

? BAD SUBS

Причина. Индекс элемента массива превышает объявленный размер.

Действие. Исправьте описание массива в команде DIMENSION.

? FORMAT

Причина. Тип константы не соответствует указанному типу.

Действие. Согласуйте тип константы с указателем типа в спецификации ячейки.

? LABEL

Причина. Недопустимая метка команды отладчика.

Действие. Переопределите макроопределение, в котором используется данная метка.

? MACRO #

Причина. Номер макроопределения вне допустимого диапазона (от 0 до 7).

Действие. Исправьте номер макроопределения.

? NO CONVERSION

Причина. В отлаживаемой программе не используются коды преобразований D, E, F и G.

Действие. Включите в загрузочный модуль отлаживаемой программы модуль RCI$, используя программу «Редактор связей».

? NO ROOM

Причина. Число пауз, установленных по команде PAUSE, превышает восемь.

Действие. Уменьшите число пауз по команде RESET.
? ONLY IN MACRO
Причина. Команда GOTO использована вне макроопределения.
Действие. Правильно используйте команду GOTO.
? PAUSE NOT FOUND
Причина. Попытка стереть паузу, установленную на несуществующий порядковый номер внутренней последовательности программы или на имя несуществующего программного модуля.
Действие. Продолжите отладку программы.
? UNDEFINED
Причина. Синтаксическая ошибка в командной строке.
Действие. Исправьте командную строку.

ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Операционная система ФОДОС-2
   Редактор связей
   Руководство оператора
2. ФОРТРАН/ФОДОС-2
   Транслятор с ФОРТРАНа
   Руководство программиста

КОНТРОЛЬНЫЕ ЗАДАЧИ. РУКОВОДСТВО ОПЕРАТОРА

1. КОНТРОЛЬНАЯ ЗАДАЧА 1

1.1. Назначение программы

Контрольная задача предназначена для проверки на работоспособность транслятора с языка ФОРТРАН и библиотеки ФОРТРАНа операционной системы ФОДОС-2 [2] под управлением монитора XM.

Проверка осуществляется на примере программы DEMO1.FOR — определения минимального значения группы чисел, с использованием виртуальных массивов.

Методика проверки включает:

1) получение объектного модуля и листинга программы;
2) получение загрузочного модуля и запуск программы.

Текст программы DEMO1.FOR приведен в приложении 1.

1.2. Условия выполнения программы

1.2.1. Требования к техническим средствам

Для выполнения программы DEMO1.FOR необходимы следующие технические средства:

— центральный процессор;
— оперативное запоминающее устройство емкостью 32K слов и диспетчер памяти для работы под управлением монитора XM;
— терминал;
— устройство ввода-вывода информации на гибких магнитных дисках или на жестком магнитном диске.

1.2.2. Требования к программным средствам

Для выполнения программы DEMO1.FOR необходимы следующие программные средства:

— компоненты операционной системы ФОДОС-2:
FMONXM.SYS — монитор системы;
SWAP.SYS — файл свопинга;
TT.SYS — драйвер терминала;
<XX>.SYS — драйвер диска;
LINK.SAV — редактор связей;

SYSLIB.OBJ — библиотека системных подпрограмм
где XX — DXX, DYX, MXX, MYX или DW.
— компоненты системы ФОРТРАН/ФОДОС-2:
FORTRA.SAV — транслятор с ФОРТРАНа;
FORLIB.OBJ — библиотека ФОРТРАНа;
DEMO1.FCR — контрольная задача 1.

1.3. Выполнение программы

1.3.1. Установка носителей

При работе с накопителем на гибких магнитных дисках установить:

— на привод 0 диск с компонентами операционной системы ФОДОС-2;

— на привод 1 диск с компонентами ФОРТРАН/ФОДОС-2 и программой DEMO1.FOR.

1.3.2. Загрузка и запуск транслятора с ФОРТРАНа

Загрузите систему ФОДОС-2 (см. кн. 1)
Подайте команду:

```
.RUN <XX>M:FORTRA
*
```

где M — номер привода с компонентами ФОРТРАН и программой DEMO1.

ПРИМЕЧАНИЯ.

1. Все команды, вводимые с терминала, должны заканчиваться нажатием клавиши <ВК>;

2. Подчеркнутый текст печатается ЭВМ.

3. Во всех случаях, если M совпадает с N, его можно не указывать.

1.3.3. Получение объектного модуля и листинга программы DEMO1

1) Введите командную строку:

```
*<XX>M:DEMO1.OBJ,TT:/L:7=<XX>M:DEMO1.FOR
*
```

На терминал выводится листинг программы, содержащий текст исходной программы, карту памяти и генерированный

нанизанный код. Листинг программы DEMO1.FOR приведен в приложении 2;

2) передайте управление монитору по команде:

```
*СУ/С
.
```

1.3.4. Получение загрузочного модуля программы DEMO1

Подайте следующие команды:

```
.R LINK
*<XX>M:DEMO1.SAV=<XX>M:DEMO1.OBJ,<XX>M:FORLIB.OBJ
*СУ/С
.
```

1.3.5. Запуск программы DEMO1

1) подайте команду:

```
.RUN <XX>M:DEMO1.SAV
CONTROL TASK
```

2) введите данные с терминала — пять целых шестизначных констант. Если число цифр в константе меньше шести, то недостающие цифры при вводе должны быть заменены пробелами.

```
      123      34       45    4456      567
ΛΛΛΛ  ΛΛΛΛ  ΛΛΛΛ  ΛΛ  ΛΛΛ
```

ПРИМЕЧАНИЕ. Символ «Λ» используется для указания пробела.

При правильном выполнении программы DEMO1 на терминал выводится сообщение:

```
MIN ELEMENT OF VECTOR L
 34
```

При неправильном выполнении программы DEMO1 на терминал выводится сообщение:

```
ERROR IN PROGRAM
MIN ELEMENT OF VECTOR L
```

3) для повторения работы программы введите с терминала новые данные;

4) для окончания работы подайте с терминала команду СУ/С.

1.4. Сообщения оператору

При работе с программой DEMO1 на терминал выводятся сообщения, текст которых приведен в п. 1.3.5.

Сообщения об ошибках, возникающих при работе с программами дисковой операционной системы ФОДОС-2, следует смотреть в документе [1] или документе на программу, при работе с которой возникла данная ошибка.

2. КОНТРОЛЬНАЯ ЗАДАЧА 2

2.1. Назначение программы

Контрольная задача предназначена для проверки на работоспособность транслятора с языка ФОРТРАН и библиотеки ФОРТРАНа операционной системы ФОДОС-2 [2] под управлением мониторов FB и SJ.

Проверка осуществляется на примере программы DEMO.FOR — определения суммы группы чисел.

Методика проверки включает:

1) получение объектного модуля и листинга программы;

2) получение загрузочного модуля и запуск программы.

Текст программы DEMO.FOR приведен в приложении 3.

2.2. Условия выполнения программы

2.2.1. Требования к техническим средствам

Для выполнения программы DEMO.FOR необходимы следующие технические средства:

— центральный процессор;
— оперативное запоминающее устройство емкостью 16К слов для работы под управлением мониторов SJ и FB;
— терминал;
— устройство ввода-вывода информации на гибких магнитных дисках или на жестком магнитном диске.

2.2.2. Требования к программным средствам

Для выполнения программы DEMO.FOR необходимы следующие программные средства:

— компоненты операционной системы ФОДОС-2:
FMON<YY>.SYS — монитор системы;
SWAP.SYS — файл свопинга;
TT.SYS — драйвер терминала;
<XX>.SYS — драйвер диска;
LINK.SAV — редактор связей;
SYSLIB.OBJ — библиотека системных подпрограмм
где YY — XM, FB или SJ;
XX — DW, MX, MY или DWX, MXX, MXY;
— компоненты системы ФОРТРАН/ФОДОС-2:
FORTRA.SAV — транслятор с ФОРТРАНа;
FORLIB.OBJ — библиотека ФОРТРАНа;
DEMO.FOR — контрольная задача 2.

2.3. Выполнение программы

2.3.1. Установка носителей

При работе с накопителем на гибких магнитных дисках установить:

— на привод 0 диск с компонентами операционной системы ФОДОС-2;
— на привод 1 диск с компонентами ФОРТРАН/ФОДОС-2 и программой DEMO.FOR.

2.3.2. Загрузка и запуск транслятора с ФОРТРАНа

Подайте команду:

.RUN <XX>M:FORTRA
—
*
—

где M — номер привода с компонентами ФОРТРАН и программой DEMO.

ПРИМЕЧАНИЯ.

1. Все команды, вводимые с терминала, должны заканчиваться нажатием клавиши <ВК>.
2. Подчеркнутый текст печатается ЭВМ.
3. Во всех случаях, если M совпадает с N, его можно не указывать.

2.3.3. Получение объектного модуля и листинга программы DEMO

1) Введите командную строку

```
*<XX>M:DEMO.OBJ,TT:/L:7=<XX>M:DEMO.FOR
—
*
—
```

На терминал выводится листинг программы, содержащий текст исходной программы, карту памяти и генерированный нанизанный код. Листинг программы DEMO.FOR приведен в приложении 4.

2) Передайте управление монитору по команде:

```
*СУ/С
—
.
—
```

2.3.4. Получение загрузочного модуля программы DEMO

Подайте следующие команды:

```
.R LINK
—
*<XX>M:DEMO.SAV=<XX>M:DEMO.OBJ,<XX>M:FORLIB
—
*СУ/С
—
.
—
```

2.3.5. Запуск программы DEMO

1) подайте команду:

```
.RUN <XX>M:DEMO.SAV
—
CONTROL TASK
------- ----
```

2) введите данные с терминала — пять целых шестизначных констант. Если число цифр в константе меньше шести, то недостающие цифры при вводе должны быть заменены пробелами.

```
      456      76        5       11      123
ΛΛΛΛ  ΛΛΛΛ  ΛΛΛΛΛ  ΛΛΛΛ  ΛΛΛ
```

ПРИМЕЧАНИЕ. Символ «Λ» используется для обозначения пробела.

При правильном выполнении программы DEMO на терминал выводится сообщение:

```
SUM OF VECTOR L
--- -- ------ -
671
-
```

При неправильном выполнении программы DEMO на терминал выводится сообщение:

```
ERROR IN PROGRAM
----- -- -------
SUM OF VECTOR L
--- -- ------ -
```

3) для повторения работы программы введите с терминала новые данные;

4) для окончания работы подайте с терминала команду СУ/С.

2.4. Сообщения оператору

При работе с программой DEMO на терминал выводятся сообщения, текст которых приведен в п. 2.3.5.

Сообщения об ошибках, возникающих при работе с программами дисковой операционной системы ФОДОС-2, следует смотреть в документе [1] или документе на программу, при работе с которой возникла данная ошибка.

ПРИЛОЖЕНИЕ 1

Текст программы DEMO1.FOR

```
      VIRTUAL L(5)
      TYPE 3
3     FORMAT (' CONTROL TASK ')
6     READ (5,1) (L(I), I=1,5)
1     FORMAT (X516)
```

```
      K=MIN0(L(1), L(2), L(3), L(4), L(5))
      I=K
      I=I/K
      IF (I.NE.1) TYPE 4
4     FORMAT ('0ERROR IN PROGRAM')
      TYPE 2
2     FORMAT (/X'MIN ELEMENT OF VECTOR L')
      WRITE (7, 1) K
      GOTO 6
      STOP
      END
```

ПРИЛОЖЕНИЕ 2

Листинг программы DEMO1.FOR

```
FORTRAN          B02.00                                    PAGE 001
0001          VIRTUAL  L(5)
0002          TYPE 3
0003  3       FORMAT (  CONTROL TASK')
0004  6       READ (5,1) (L(I),I=1,5)
0005  1       FORMAT (X5I6)
0006          K=MIN0 (L(1),L(2),L(3),L(4),L(5))
0007          I=K
0008          I=I/K
0009          IF (I.NE.1) TYPE 4
0011  4       FORMAT ('0ERROR IN PROGRAM')
0012          TYPE 2
0013  2       FORMAT (/X'MIN ELEMENT OF VECTOR L')
0014          WRITE (7,1) K
0015          GOTO 6
0016          STOP
0017          END
FORTRAN           STORAGE MAP FOR PROGRAM UNIT .MAIN.
LOCAL VARIABLES, .PSECT $DATA, SIZE = 000020 (     8. WORDS)
NAME  TYPE OFFSET  NAME  TYPE  OFFSET   NAME   TYPE  OFFSET
I      I*2   000000   K     I*2   000002
VIRTUAL ARRAYS, TOTAL SIZE = 00000100 (      32. WORDS)
NAME      TYPE   OFFSET     --------SIZE--------- DIMENSIONS
L       I*2     00000000   00000012 (      5.) (5)
SUBROUTINES,FUNCTIONS,STATEMENT AND PROCESSOR-DEFINED
FUNCTIONS:
NAME  TYPE  NAME  TYPE   NAME  TYPE  NAME  TYPE  NAME  TYPE
MINO   I*2
FORTRAN           GENERATED CODE FOR PROGRAM UNI.MAIN.
 STATEMENT #0002
 000006           LSN$     #000002
 000012           REL$     $DATAP+#000120
 000016           REL$     $DATAP+#000010
 000022           IFW$
```



```
 000024           EOL$
STATEMENT -#0004
 000026           LSN$     #000004
 000032           REL$     $DATAP+#000116
 000036           REL$     $DATAP+#000031
 000042           IFR$
 000044           MOI$1M   $DATA+#000000
 000050           REL$     $DATA+#000004
 000054           TVI$
 000056           MOI$MS   $DATA+#000000
 000062           DCI$S
 000064           VSI$MM   $DATA+#000004     .VIR.+#000000
 000072           NMI$1I   $DATA+#000000     #000005 000050
 000102           EOL$
STATEMENT #0006
 000104           LSN$     #000006
 000110           VGI$IM   #000004 .VIR.+#000000
 000116           MOI$SM   $DATA+#000006
 000122           REL$     $DATA+#000006
 000126           VGI$IM   #000003 .VIR.+#000000
 000134           MOI$SM   $DATA+#000010
 000140           REL$     $DATA+#000010
 000144           VGI$IM   #000002 .VIR.+#000000
 000152           MOI$SM   $DATA+#000012
 000156           REL$     $DATA+#000012
 000162           VGI$IM   #000001 .VIR.+#000000
 000170           MOI$SM   $DATA+#000014
 000174           REL$     $DATA+#000014
 000200           VGI$IM   #000000 .VIR.+#000000
 000206           MOI$SM   $DATA+#000016
 000212           REL$     $DATA+#000016
 000216           CAL$     #000005 MIN0+#000000
 000224           MOI$RM   $DATA+#000002
STATEMENT #0007
 000230           ISN$
 000232           MOI$MM   $DATA+#000002     $DATA+#000000
STATEMENT #0008
 000240           ISN$
 000242           MOI$MS   $DATA+#000000
 000246           DII$MS   $DATA+#000002
 000252           MOI$SM   $DATA+#000000
STATEMENT #0009
 000256           ISN$
 000260           CMI$MI   $DATA+#000000    #000001
FORTRAN           GENERATED CODE FOR PROGRAM UNIT .MAIN.
 000266           BEG$     000310
STATEMENT #0010
 000272           ISN$
 000274           REL$     $DATAP+#000120
 000300           REL$     $DATAP+#000036
 000304           IFW$
 000306           EOL$
STATEMENT #0012
 000310           LSN$     #000014
 000314           REL$     $DATAP+#000120
```

```
000320          REL$    $DATAP+#000062
000324          IFW$
000326          EOL$
STATEMENT #0014
000330          LSN$    #000016
000334          REL$    $DATAP+#000120
000340          REL$    $DATAP+#000031
000344          IFW$
000346          REL$    $DATA+#000002
000352          TVI$
000354          EOL$
STATEMENT #0015
000356          ISN$
000360          BRA$    000026
```

ПРИЛОЖЕНИЕ 3

Текст программы DEMO.FOR

```
      DIMENSION L(5)
      TYPE 3
3     FORMAT (' CONTROL TASK')
6     READ (5,1) (L(I),I=1,5)
1     FORMAT (X5I6)
      K=(L(1)+L(2)+L(3)+L(4)+L(5))
      I=K
      I=I/K
      IF (I.NE.1) TYPE 4
4     FORMAT ('0ERROR IN PROGRAM')
      TYPE 2
2     FORMAT (/X'SUM OF VECTOR L')
      TYPE 2
      WRITE (7,1) K
      GOTO 6
      STOP
      END
```

ПРИЛОЖЕНИЕ 4

Листинг программы DEMO.FOR

```
FORTRAN         B02.00                                  PAGE 001
0001          DIMENSION L(5)
0002          TYPE 3
0003    3     FORMAT (' CONTROL TASK')
0004          READ (5,1) (L(I),I=1,5)
0005    1     FORMAT (X5I6)
0006          K=(L(1)+L(2)+L(3)+L(4)+L(5))
0007          I=K
0008          I=I/K
0009          IF (I.NE.1) TYPE 4
0011    4     FORMAT ('0ERROR IN PROGRAM')
0012          TYPE 2
0013    2     FORMAT (/X'SUM OF VECTOR L')
0014          WRITE (7,1) K
0015          GOTO 6
0016          STOP
0017          END
```



```
FORTRAN         STORAGE MAP FOR PROGRAM UNIT .MAIN.
LOCAL VARIABLES, .PSECT $DATA, SIZE = 000020 (     8. WORDS)
NAME   TYPE  OFFSET    NAME   TYPE  OFFSET   NAME   TYPE  OFFSET
I       I*2  000014    K       I*2  000016
LOCAL AND COMMON ARRAYS:
NAME      TYPE    SECTION OFFSET  ------SIZE----- DIMENSIONS
          I*2       $DATA   000000  000012 (     5.) (5)
SUBROUTINES,FUNCTIONS,STATEMENT AND PROCESSOR-DEFINED FUNCTIONS:
NAME   TYPE  NAME   TYPE   NAME   TYPE   NAME   TYPE   NAME   TYPE
MINO   I*2
FORTRAN         GENERATED CODE FOR PROGRAM UNIT .MAIN.
 STATEMENT #0002
 000006          LSN$    #000002
 000012          REL$    $DATAP+#000120
 000016          REL$    $DATAP+#000010
 000022          IFW$
 000024          EOL$
 STATEMENT #0004
 00002          LSN$    #000004
 000032          REL$    $DATAP+#000116
 000036          REL$    $DATAP+#000031
 000042          IFR$
 000044          MOI$IM  $DATA+#000014
 000050          SAI$IM  #000001 $DATA+#177776
 000056          MOI$SM  $DATA+#000012
 000062          MOI$MS  $DATA+#000012
 000066          TVI$
 000070          ADI$IM  #000002 $DATA+#000012
 000076          NHI$1I  $DATA+#000014  #000005 000062
 000106          EOL$
 STATEMENT #0006
 000110          LSN$    #000006
 000114          REL$    $DATA#000010
 000120          REL$    $DATA#000006
 000124          REL$    $DATA#000004
 000130          REL$    $DATA#000002
 000134          REL$    $DATA#000000
 000140          CAL$    #000005 MIN0+#000000
 000146          MOI$RM  $DATA+#000016
 STATEMENT #0007
 000152          ISN$
 000154          MOI$MM  $DATA+#000016  $DATA+#000014
 STATEMENT #0008
 000162          ISN$
 000164          MOI$MS  $DATA+#000014
 000170          DII$MS  $DATA+#000016
 000174          MOI$SM  $DATA+#000014
 STATEMENT #0009
 000200          ISN$
 000202          CMI$MI  $DATA+#000014  #000001
 000210          BEQ$    000232
 STATEMENT #0010
 000214          ISN$
 000216          REL$    $DATA+#000120
 000222          REL$    $DATA+#000036
```

```
000226          IFW$
000230          EOL$
STATEMENT #0012
FORTRAN         GENERATED CODE FOR PROGRAM UNIT .MAIN.
000232          LSN$    #000014
000236          REL$    $DATAP+#000120
000242          REL$    $DATAP+#000062
000246          IFW$
000250          EOL$
STATEMENT #0014
000252          LSN$    #000016
000256          REL$    $DATAP+#000120
000262          REL$    $DATAP+#000031
000266          IFW$
000270          REL$    $DATA+#000016
000274          TVI$
000276          EOL$
STATEMENT #0015
000300          ISN$
000302          BRA$    000026
```

ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Операционная система ФОДОС-2
   Сообщения системы
2. ФОРТРАН/ФОДОС-2
   Описание применения

СИСТЕМА УПРАВЛЕНИЯ ГРАФИЧЕСКИМ ТЕРМИНАЛОМ

ГРАФИЧЕСКИЙ ПАКЕТ. РУКОВОДСТВО ПРОГРАММИСТА.

1. НАЗНАЧЕНИЕ, УСЛОВИЯ ПРИМЕНЕНИЯ И ХАРАКТЕРИСТИКИ ПАКЕТА ПРОГРАММ

Настоящая документация предназначена для ознакомления с системой управления графическим терминалом и позволяет программисту писать программы на языке ФОРТРАН, используя библиотеку графических подпрограмм.

Работа графической библиотеки осуществляется под управлением SJ— или FB— мониторов.

Условия, необходимые для обеспечения графического пакета языка высокого уровня:

— наличие платы контроллера графического дисплея (КГД);
— память не менее 12К слов;
— системный терминал;
— накопитель на гибком магнитном диске или другое системное устройство внешней памяти с прямым доступом.

Объем памяти необходимый для работы графического пакета, зависит от конкретных требований, но он не может быть менее 12К слов.

Графические подпрограммы позволяют выполнить следующие графические функции с помощью простых операторов CALL из программ:

— отображение точек, векторов, текста и графических данных;
— вращение изображения;
— масштабирование изображения для любой координатной системы;
— независимое управление частями изображения;
— вычерчивание пунктирных линий;
— установка одной из трех координатных систем: линейная, логарифмическая, полярная.

2. ОБРАЩЕНИЕ К ПОДПРОГРАММАМ ПАКЕТА, ВХОДНЫЕ И ВЫХОДНЫЕ ПАРАМЕТРЫ

Обращение к подпрограммам пакета из ФОРТРАН-программы выполняется одинаково для всех программ с использованием оператора CALL, с указанием имени программы, к которой производится обращение и предварительной засылкой фактических параметров в массив параметров и адреса этого массива в регистр R5. Формат обращения:
CALL <ИМЯ ПРОГРАММЫ>
Имя программы должно быть определено глобальным в основной программе, содержащей операторы вызова. Формат обращения к конкретным подпрограммам и требуемые входные и выходные параметры этих подпрограмм приведены при описании каждой подпрограммы.

3. СООБЩЕНИЯ

В процессе выполнения программ пакета на экран монитора могут быть выведены различные сообщения. Сюда относятся сообщения об ошибках, о ходе вычислительного процесса и т. д.

Каждое сообщение имеет стандартный вид:
<ИМЯ СООБЩЕНИЯ> <ТИП СООБЩЕНИЯ>

4. ИНИЦИАЛИЗАЦИЯ И АБСОЛЮТНАЯ ГРАФИКА

4.1. Инициализация. Подпрограмма INITT

При использовании программ системы управления терминалом в первую очередь следует осуществить инициализацию терминала и области состояния терминала. Это может быть сделано подпрограммой INITT.

При вызове подпрограммы INITT производятся следующие действия:
1) экран очищается и алфавитно-цифровой курсор передвигается в исходное положение в левый нижний угол экрана;
2) терминал устанавливается в алфавитно-цифровой режим;
3) ширина экрана определяется его левой и правой физическими границами (X:0—399; Y:0—279);
4) окно определяется так, что может быть выведена часть виртуального пространства, эквивалентная по координатам экрану (т. е. точка (27,76) в координатах пользователя эквивалентна точке (27,76) в координатах экрана).

Подпрограмма INITT требует задать в качестве входного

параметра скорость передачи символов между ЭВМ и терминалом.
Форма обращения:
CALL INITT (IBAUD)
Входной параметр:
IBAUD — скорость передачи (бод) в символах в секунду

4.2. Окончание. Подпрограмма FINITT

При окончании программы, использующей систему управления терминалом, желательно вернуть терминал в алфавитно-цифровой режим и отключить графику.

Подпрограмма FINITT автоматически выполняет эти функции. Она заканчивает программу и отключает графику. Ее аргумент определяет позицию графического луча при завершении программы. Форма обращения:
CALL FINITT (IX, IY)
Входные параметры:
IX — X — координата позиции луча на экране при завершении
IY — Y — координата позиции луча на экране при завершении.

4.3. Вычерчивание абсолютных векторов в координатах экрана

Прямые линии и точки относительно экранных координат вычерчивают три подпрограммы: MOVABS, DRWABS и PNTABS. Сочетание «ABS» обозначает абсолютный формат; черчение называется абсолютным потому, что отсчет ведется от фиксированной точки — начала координат (0,0). Аргументы этих процедур всегда целого типа.

4.3.1. Подпрограмма MOVABS

Аргументами подпрограммы MOVABS является пара координат точки, в которую нужно переместить луч на экране. Координаты предыдущего положения луча сохраняются как текущее положение луча. Эти значения изменяются каждый раз после вычерчивания линии или другой команды вывода. Форма обращения:
CALL MOVABS (IX, IY)
Пример:
CALL MOVABS (100, 150)

Такое обращение вызывает перемещение луча в точку (100, 150), и последующее черчение может начаться с этой точки.

4.3.2. Подпрограмма DRWABS

Подпрограмма генерирует видимый вектор из текущего положения луча в точку с заданными координатами и соот-

ветственно изменяет переменные в области состояний терминала.

Форма обращения:

CALL DRWABS (IX, IY)

Пример:

```
CALL MOVABS (100, 50)
CALL DRWABS (300, 50)
```

Такая последовательность обращений вызывает перемещение луча в точку (100, 50) и последующее вычерчивание линии от точки (100, 50) до точки (300, 50).

Пример: программа вычерчивания треугольника (рис. 1).

```
CALL INITT (30)
CALL MOVABS (100, 100)
CALL DRWABS (300, 100)
CALL DRWABS (200, 187)
CALL DRWABS (100, 100)
CALL FINITT (0, 67)
CALL H
```

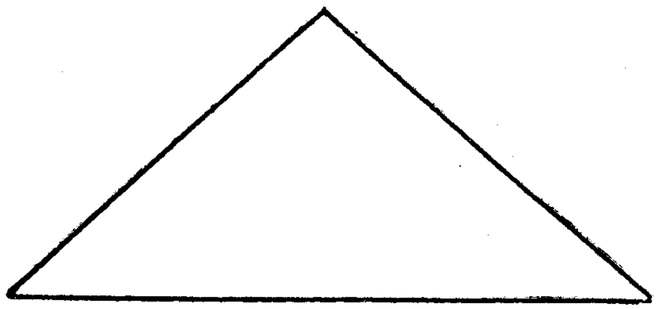


Рис. 1

4.3.3. Подпрограмма PNTABS

Подпрограмма перемещает луч в точку с координатами, заданными как параметры, и высвечивает ее.

Форма обращения: CALL PNTABS (IX, IY)

Пример: программа вычерчивания квадрата (рис. 2) с точкой в центре.

```
CALL INITT (30)
CALL MOVABS (100, 100)
CALL DRWABS (100, 200)
CALL DRWABS (200, 200)
CALL DRWABS (200, 100)
CALL DRWABS (100, 100)
CALL PNTABS (150, 150)
CALL FINITT (0, 67)
CALL H
```

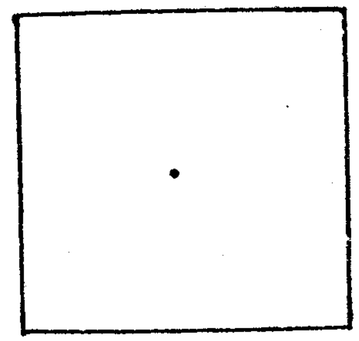



Рис. 2

4.4. Вычерчивание линий по приращениям экранных координат

Часто проще начертить линию, указав, на сколько единиц нужно сдвинуться по горизонтали и по вертикали относительно последнего положения луча. Отрицательное относительное движение вызывает сдвиг влево или вниз, когда положительное сдвигает вправо или вверх. Подпрограммы DRWREL, MOVREL и PNTREL выполняют относительное черчение по приращениям, заданным в единицах экрана. Они имеют такой же синтаксис, как DRWABS, PNTABS и MOVABC. Выполнение этих программ соответствует действиям, приведенным в табл. 1.

Таблица 1

СВОДНАЯ ТАБЛИЦА ПОДПРОГРАММ

Экранная графика (целые аргументы)		
1	2	3
Действие	Абсолютный	Относительный
Сдвиг	MOVABS	MOVREL
Черчение	DRWABS	DRWREL
Точка	PNTABS	PNTREL

Пример: начертить квадрат (рис. 2) с использованием относительных векторов.

```
CALL INITT (30)
CALL MOVABS (100, 100)
```

```
CALL DRWREL (100,0)
CALL DRWREL (0,100)
CALL DRWREL (—100,0)
CALL DRWREL (0, —100)
CALL PNTREL (50,50)
CALL FINITT (0,67)
CALL H
```

ПРИМЕЧАНИЕ

Подпрограмма H предназначена для получения твердой графической копии на печать.

5. ВИРТУАЛЬНАЯ И ЭКРАННАЯ ГРАФИКА

В этом разделе обсуждаются самые сложные отношения в системе управления терминалом: перевод данных пользователя в физические точки на экране. Понимая соотношения между областью данных и экраном терминала, пользователь может свободно управлять выводом на экран результатов своих расчетов. Например, он может вывести на экран одновременно три графика, отображающих различные наборы данных.

В подразделах 5.1—5.5 этого раздела рассматривается вывод области данных пользователя. Эта область представляется виртуально существующей внутри ЭВМ и аналогична листу бумаги, на котором просто рисуются графические данные. Область данных называется виртуальным пространством. Единица измерения в виртуальном пространстве произвольна и может быть выражена в любых желаемых единицах, от миллиграммов до световых лет.

В подразделах 5.6—5.7 этого раздела объясняется, как виртуальные данные могут быть отображены на определенных участках экрана терминала (экранное окно). В подразделах 5.8—5.9 особо рассматривается взаимозависимость графики в виртуальном пространстве и экранной графики. В подразделах 5.10 и 5.11 обсуждается черчение пунктирных линий как в виртуальном пространстве, так и в экранных координатах.

5.1. Виртуальное окно

Виртуальное пространство полностью или любую его часть можно рассматривать как окно. Пользователь описывает в своих единицах прямоугольник — виртуальное окно, определяющее часть виртуального пространства, предполагаемого

для вывода на экран. Все отрезки прямых линий (векторы) или их части, лежащие вне виртуального окна, автоматически уничтожаются или отсекаются графическими процедурами, в то время как лежащие внутри или проходящие через окно масштабируются и переносятся в соответствующую часть экрана.

5.2. Видимая область в координатах пользователя (виртуальное окно)

Пусть имеем таблицу количества станков, производимых заводом в течении десяти дней. Рассмотрим способ вывода табличных данных на экран. Для примера возьмем следующую последовательность данных:

День	Число станков
1	30
2	26
3	42
4	38
5	40
6	50
7	54
8	48
9	40
10	52

Виртуальное окно (рис. 3) определяется в виртуальном пространстве с координатами нижнего левого угла в точке: 0 дней — 20 станков. По горизонтальной оси X окно простирается на 10 дней, а по вертикальной оси Y — на 40 станков. Одним из способов описания прямоугольного окна может быть задание положения нижнего левого угла и горизонтального и вертикального размеров.

Данные при выводе на экран масштабируются пропорционально расстояниям в виртуальном окне. Так как ширина экрана составляет 400 единиц, то каждая следующая точка будет выводиться на экран через (1/10)*400=40 единиц растра. Точка (1.30) располагается на 1/4 высоты виртуального окна. Поскольку экран содержит 280 единиц растра по высоте, то точка будет отстоять от нижнего края на (1/4)*280= =70 единиц (рис. 4). Все остальные точки, лежащие в пределах виртуального окна пересчитываются в координаты точек экрана аналогично.

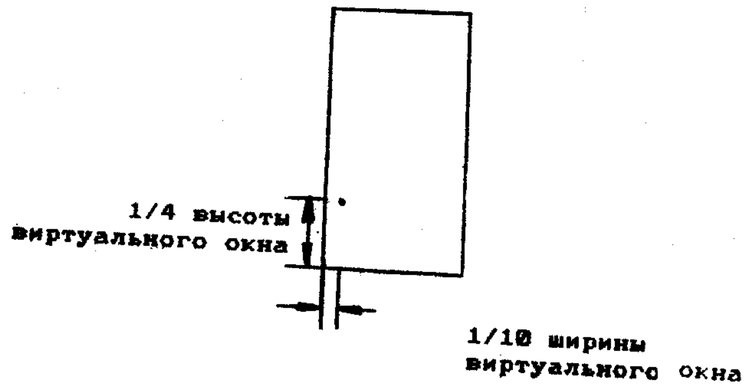


Рис. 3

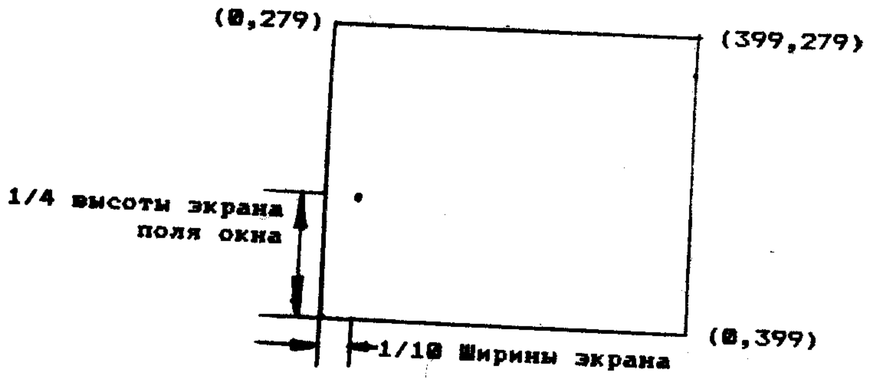


Рис.4

5.3. Определение виртуального окна

Система управления графическим терминалом использует одну из двух подпрограмм определения виртуального окна: VWINDO и DWINDO. Различие между ними заключается в том, что в подпрограмме VWINDO в качестве параметров задаются начальная точка окна и размеры, а в DWINDO задаются координаты нижнего левого и верхнего правого угла.

5.3.1. Подпрограмма VWINDO

Форма обращения:

CALL VWINDO(XMIN,XRANGE,YMIN,YRANGE)

Входные параметры:

XMIN — минимальная горизонтальная координата пользователя;

XRANGE — горизонтальный размер прямоугольника;

YMIN — минимальная вертикальная координата пользователя;

YRANGE — вертикальный размер прямоугольника;

Пример: форма обращения для определения виртуального окна, приведенного на рис. 3:

CALL VWINDO(0.,10.,20.,40.)

5.3.2. Подпрограмма DWINDO

Второй метод определения виртуального окна — использование подпрограммы DWINDO.

Форма обращения:

CALL DWINDO(XMIN,XMAX,YMIN,YMAX)

Входные параметры:

XMIN — минимальная горизонтальная координата пользователя;

XMAX — максимальная горизонтальная координата пользователя;

YMIN — минимальная вертикальная координата пользователя;

YMAX — максимальная вертикальная координата пользователя;

Пример: форма обращения для определения виртуального окна, приведенного на рис. 3:

CALL DWINDO(0.,10.,20.,60.)

5.4. Вычерчивание линий в координатах пользователя (виртуальных) по абсолютным значениям координат

Подпрограммы MOVEA, DRAWA и POINTA аналогичны подпрограммам MOVABS, DRWABS и PNTABS, но они позволяют задавать в качестве параметров координаты точек вне виртуального окна.

При этом выведены будут только те точки или части видимых векторов (отрезки прямых), которые попадают внутрь границ окна; эта операция называется «отсечение».

Форма обращения:

CALL MOVEA(X,Y)
CALL DRAWA(X,Y)
CALL POINTA(X,Y)

Входные параметры:

X — виртуальная (вещественная) горизонтальная координата точки, в которую будет перемещен луч или проведен вектор или выведена точка.

Y — виртуальная (вещественная) вертикальная координата точки, в которую будет перемещен луч или проведен вектор или выведена точка.

Пример: вывод данных, приведенных на рис. 4

```
      CALL INITT(30)
      CALL VWINDO(0.,10.,20.,40.)
      DIMENSION X(10), Y(10)
      DATA X/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./
      DATA Y/30.,26.,42.,38.,40.,50.,54.,48.,40.,52./
      DO 10 I=1,10
10    CALL POINTA (X(I), Y(I))
      CALL FINITT (0,67)
      CALL H
```

Для работы в координатах пользователя на экране можно установить виртуальное окно. В следующем примере вычерчивается прямоугольник размером 3×3 единиц с размещением левого угла в точке (1,1) и точкой в центре (рис. 5).

Пример:

```
      CALL INITT(30)
      CALL VWINDO(0.,8.,0.,6.)
      CALL MOVEA(1.,1.)
      CALL DRAWA(1.,4.)
      CALL DRAWA(4.,4.)
      CALL DRAWA(4.,1.)
      CALL DRAWA(1.,1.)
      CALL POINTA(2.5,2.5)
      CALL FINITT(0,67)
      CALL H
```

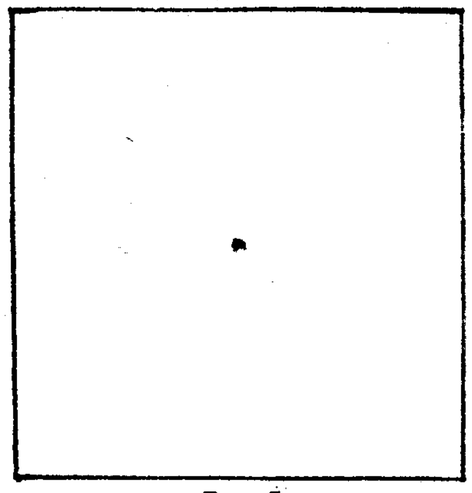


Рис. 5

5.5. Подпрограммы для относительных виртуальных координат

Подпрограммы MOVER, DRAWR, POINTR выполняют перемещение луча, черчение отрезка и высвечивание точки, соответственно, относительно текущего положения луча. Они аналогичны подпрограммам MOVREL, DRWREL и PNTREL за исключением того, что оперируют с координатами пользователя, а не экрана. Для этих подпрограмм также реализована операция отсечения. Выполнение этих подпрограмм соответствует действиям, приведенным в табл. 2.

В следующем примере рисуется такой же прямоугольник, как на рис. 5.

Пример:

```
      CALL INITT(30)
      CALL VWINDO(0.,8.,0.,6.)
      CALL MOVER(1.,1.)
      CALL DRAWR(0.,3.)
      CALL DRAWR(3.,0.)
      CALL DRAWR(0.,—3.)
      CALL DRAWR(—3.,0.)
      CALL POINTR(1.5,1.5)
      CALL FINITT(0,67)
      CALL H
```

Таблица 2

СВОДНАЯ ТАБЛИЦА НАЗВАНИЙ ПОДПРОГРАММ

	Экранная графика целые аргументы		Виртуальная графика вещественные аргументы	
1	2		3	
Действие	Абсолютный	Относительный	Абсолютный	Относительный
Сдвиг	MOVABS	MOVREL	MOVEA	MOVER
Черчение	DRWABS	DRWREL	DRAWA	DRAWR
Точка	PNTABS	PNTREL	POINTA	POINTR

5.6. Экранное окно

До сих пор для вывода чертежа в виртуальном пространстве использовался весь экран. Но в качестве области вывода может быть задана любая прямоугольная часть экрана. Эта область вывода называется экранным окном и определяется подпрограммами SWINDO и TWINDO. Обе подпрограммы имеют такое же отношение друг к другу, как и подпрограммы VWINDO и DWINDO. Поскольку эти подпрограммы ра-

ботают в экранных координатах, аргументы подпрограмм SWINDO и TWINDO имеют целый тип.

5.6.1. Подпрограмма SWINDO

Форма обращения:

CALL SWINDO (MINX,LENX,MINY,LENY)

Входные параметры:

MINX — минимальная экранная горизонтальная координата
LENX — горизонтальный размер прямоугольника
MINY — минимальная экранная вертикальная координата
LENY — вертикальный размер прямоугольника

5.6.2. Подпрограмма TWINDO

Форма обращения:

CALL TWINDO (MINX,MAXX,MINY,MAXY)

Входные параметры:

MINX — минимальная экранная горизонтальная координата
MAXX — максимальная экранная горизонтальная координата
MINY — минимальная экранная вертикальная координата
MAXY — максимальная экранная вертикальная координата

5.7. Масштабирование и растягивание экранного окна

Координаты точек виртуального окна масштабируются для размещения в пределах экранного окна совершенно аналогично ранее рассмотренному случаю полного экрана (см. подраздел 5.2 и рис. 3).

Подпрограммы следующего примера иллюстрируют возможности управления размером и формой экранного окна путем изменения параметров подпрограммы TWINDO. Следует отметить, что виртуальные данные полностью выводятся в каждом случае (см. рис. 6).

Пример:

```
C * демонстрация масштабирования и растягивания
      CALL INITT(30)
      CALL DWINDO(0.,10.,20.,60.)
C * вычерчивание чертежа в трех терминальных окнах
      CALL TWINDO (0,100,150,279)
      CALL GRAFIT
      CALL TWINDO(200,399,150,279)
      CALL GRAFIT
      CALL TWINDO(0,100,0,150)
      CALL GRAFIT
C * твердая копия
      CALL HDCOPY
      CALL FINITT(0,67)
      CALL H
      END
```



```
C * подпрограмма черчения графика, заполняющего терминальное C * окно
      SUBROUTINE GRAFIT
      DIMENSION X(10), Y(10)
      DATA X/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./
      DATA Y/30.,26.,42.,38.,40.,50.,54.,48.,40.,52./
      CALL MOVEA(X(1),Y(1))
      DO 10 I=1,10
10    CALL DRAWA(X(I),Y(I))
      CALL MOVEA(0.,20.)
      CALL DRAWA(10.,20.)
      CALL DRAWA(10.,60.)
      CALL DRAWA(0.,60.)
      CALL DRAWA(0.,20.)
      RETURN
      END
```

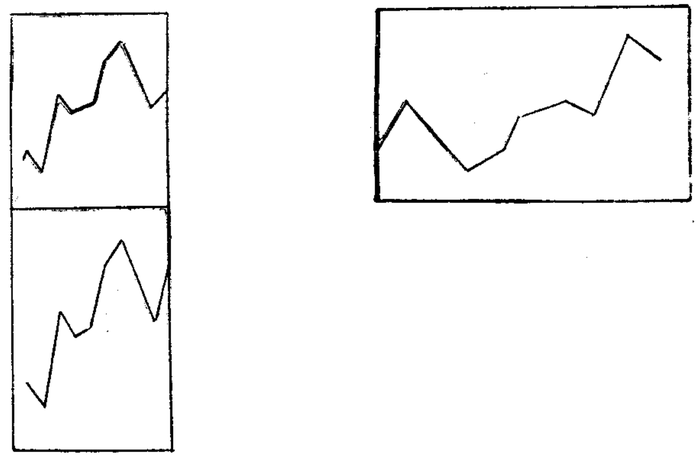


Рис. 6

5.8. Отсечение чертежа в виртуальном пространстве

Чтобы увидеть часть данных пользователь может изменить параметры подпрограмм VWINDO, DWINDO, включив только выбранную часть. Когда чертеж определен в координатах пользователя или в виртуальных координатах, то часть чертежа, оказавшаяся вне текущего виртуального окна отсекается. Операция отсечения выполняется всегда при работе с виртуальной графикой.

Пример: изображение части данных приведенных в подразделе 5.2. (см. рис. 7).

```
      DIMENSION X(10), Y(10)
      DATA X/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./
      DATA Y/30.,26.,42.,38.,40.,50.,54.,48.,40.,52./
      CALL INITT(30)
      CALL SWINDO(0,300,0,250)
      CALL VWINDO(3.,5.,40.,15.)
      CALL MOVEA(X(1),Y(1))
      DO 10 I=1,10
10    CALL DRAWA(X(I),Y(I))
      CALL MOVABS(0,0)
      CALL DRWABS(300,0)
      CALL DRWABS(300,250)
      CALL DRWABS(0,250)
      CALL DRWABS(0,0)
      CALL FINITT(0,67)
      CALL H
      STOP
      END
```

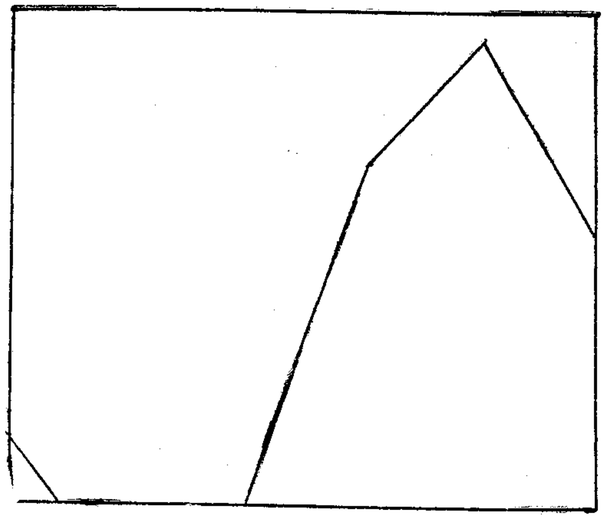


Рис. 7

5.9. Взаимозаменяемость виртуальной и экранной графики

Пользователь может использовать точки виртуального пространства, не лежащие внутри пределов виртуального окна. Он может свободно использовать координаты такой точки для определения линий чертежа, поскольку при черчении данные будут автоматически масштабированы и часть чертежа, попадающая за пределы виртуального окна будет отсечена. Но это свойство не сохраняется для экранного пространства,

которое полностью определяется границами экрана.

Преобразование экранного пространства в виртуальное возможно всегда, обратно — нет. Если точка, заданная в виртуальном пространстве, не попадет в пределы экранного окна, то при использовании экранных координат линия начнется с последней видимой позиции луча внутри экранного окна, а не с ожидаемой виртуальной точки. Кроме этого пользователь должен учитывать возможность «перескакивания» точки на другую сторону экрана, если происходит обращение к точке, лежащей вне экрана.

Пример:

Задание экранных координат, равных (500,0) вызовет «перескок» (например, при относительном вычерчивании будет получен вектор вдоль оси X длиной (500—399) единиц растра, начиная с текущей позиции луча) (см. рис. 8).

5.10. Вычерчивание пунктирных линий

Имеются 4 основных подпрограммы вычерчивания пунктирных линий: DSHABS, DSHREL, DASHA и DASHR. Эти подпрограммы аналогичны подпрограммам DRWABS, DRWREL, DRAWA и DRAWR; каждая подпрограмма вычерчивания пунктирной линии имеет третий параметр (L) целого формата. Этот параметр указывает тип пунктирной линии и может принимать целые значения от —1 до 32767. Спецификация этого параметра приведена в п. 5.11. Выполнение этих подпрограмм соответствует действиям, приведенным в табл. 3.

5.10.1. Подпрограмма DASHA

Пунктирная линия проводится из последней точки, где находился луч на экране, в заданную точку с помощью подпрограммы DASHA. Видимой будет только та часть линии, которая попадает в пределы виртуального окна. После возврата из подпрограммы луч остается в точке, заданной виртуальными координатами в обращении к подпрограмме. Форма обращения:

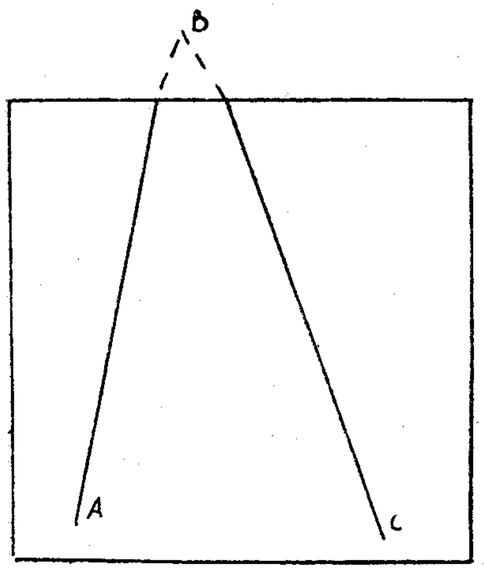
CALL DASHA (X,Y,L)

Входные параметры:

X — виртуальная X — координата точки
Y — виртуальная Y — координата точки
L — параметр пунктирной линии

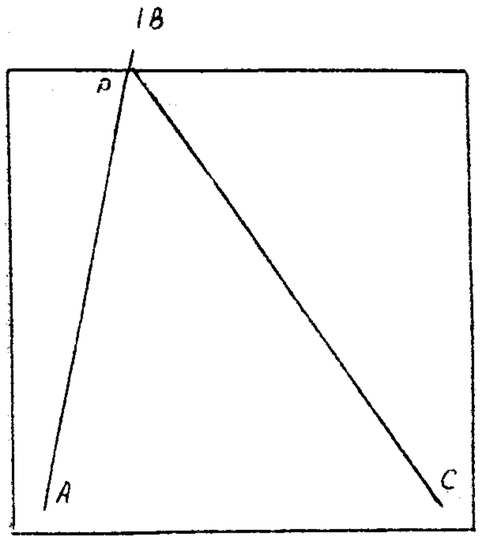
5.10.2. Подпрограмма DASHR

При использовании подпрограммы DASHR пунктирная линия может быть проведена из текущего положения луча в виртуальном пространстве в точку, отстоящую от текущей



Экранное окно

Отрезки прямой линии в координатах пользователя (виртуальных) от А к В и обратно к С



Отрезки прямой линии в виртуальных координатах от точки А к В и от точки Р к С в экранных координатах. Точка расположена вне экранного окна, но ее координаты находятся внутри границ окна

Изображение ошибки пользователя

Рис. 8

точки на величину X и Y. Отображаться будет только та часть линии, которая попадет в пределы виртуального окна.

Форма обращения:

CALL DASHR(X,Y,L)

Входные параметры:

X — значение перемещения по оси X
Y — значение перемещения по оси Y
L — параметр пунктирной линии

5.10.3. Подпрограмма DSHABS

С помощью подпрограммы DSHABS пунктирная линия может быть проведена из текущего положения луча в любую точку на экране. После возврата из подпрограммы луч останется в точке, определенной заданными экранными координатами.

Форма обращения:

CALL DSHABS(X,Y,L)

Входные параметры:

X — экранная X — координата заданной точки
Y — экранная Y — координата заданной точки
L — параметр пунктирной линии

5.10.4. Подпрограмма DSHREL

Пунктирная линия может быть начерчена на экране относительно текущего положения луча в соответствии с заданными значениями перемещения по осям X и Y в обращении к подпрограмме. Форма обращения:

CALL DSHREL(IX,IY,L)

Входные параметры:

IX — перемещение по оси X
IY — перемещение по оси Y
L — параметр пунктирной линии

5.11. Спецификация пунктирной линии: параметр L

Вид пунктирной линии задается связанными цифрами в обращении к подпрограмме, что определяет длину штриха и видимость линии. Все коды, за исключением 9, должны содержать 2 или больше цифр.

1 — Длина штриха в 5 растровых единиц
2 — Пробел в 5 растровых единиц
3 — Длина штриха в 10 растровых единиц
4 — Пробел в 10 растровых единиц
5 — Длина штриха в 25 растровых единиц
6 — Пробел в 25 растровых единиц
7 — Длина штриха в 50 растровых единиц
8 — Пробел в 50 растровых единиц
1 — Вызывает перемещение луча без вычерчивания
0 — Вызывает черчение
9 — Попеременное черчение штрихов и пробелов при вычерчивании линии между заданными точками.

Пример:

CALL DSHABS(200,100,3454)
CALL DSHABS(200,200,—1)

Следующий ниже пример иллюстрирует вычерчивание двух программно-смоделированных типов пунктирных линий (2 и 3). Здесь для выполнения относительного черчения исполь-

зуется функция KIN, переводящая дюймы в экранные координаты (см. рис. 9).

Пример:

```
C * пример вычерчивания фланца
      CALL INITT(30)
      CALL MOVABS(100,10)
      CALL DRWREL(0,KIN(5.))
      CALL DRWREL(KIN(0.5),0)
      CALL DRWREL(0,—KIN(5.))
      CALL DRWREL(—KIN(0.5),0)
      CALL MOVREL(KIN(0.5),KIN(1.5))
      CALL DRWREL(KIN(2.0),0)
      CALL DRWREL(0,KIN(2.0))
      CALL DRWREL(—KIN(2.0),0)
      CALL MOVREL(KIN(2.0), —KIN(0.25))
C * черчение пунктирных скрытых линий
      CALL DSHREL(—KIN(2.5),0.3))
      CALL MOVREL(KIN(2.5), —KIN(1.5))
      CALL DSHREL(—KIN(2.5),0.3)
      CALL MOVREL(KIN(3.0),KIN(0.75))
C * черчение штрих — пунктирной центральной оси
      CALL DSHREL(—KIN(3.5),0.2)
      CALL FINITT(0,0)
      CALL H
      END
```

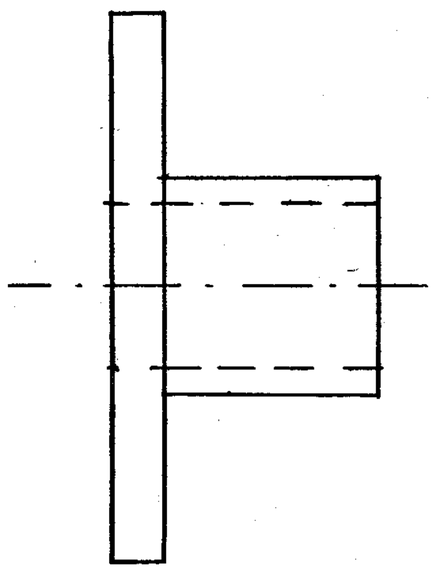


Рис. 9

Таблица 3

СВОДНАЯ ТАБЛИЦА НАЗВАНИЙ ПОДПРОГРАММ

	Экранная графика целые аргументы		Виртуальная графика вещественные аргументы	
1	2		3	
Действие	Абсолютный	Относительный	Абсолютный	Относительный
Сдвиг	MOVABS	MOVREL	MOVEA	MOVER
Черчение	DRWABS	DRWREL	DRAWA	DRAWR
Точка	PNTABS	PNTREL	POINTA	POINTR
Пунктир	DSHABS	DSHREL	DASHA	DASHR

6. СЛУЖЕБНЫЕ ПРОЦЕДУРЫ

6.1. Алфавитно-цифровой вывод

Система управления терминалом позволяет выводить строки алфавитно-цифровых (А/Ц) символов, не пользуясь операторами ФОРТРАНа READ и WRITE. Такой режим упрощает контроль за состоянием терминала, что особенно важно при слежении за текущим положением луча. Такое слежение необходимо при управлении табулятором и при контроле выхода луча за края экрана, особенно при комбинированном выводе А/Ц и графической информации. В системе управления терминалом осуществляется обязательная буферизация как графической, так и А/Ц выводимой информации. Содержимое буфера сохраняется до тех пор, пока не будет вызвана подпрограмма вывода буфера или буфер не заполнится.

6.2. Вход в алфавитно-цифровой режим. Подпрограмма ANMODE

Иногда у пользователя возникает необходимость вывести алфавитно-цифровые (А/Ц) данные иначе чем через систему управления терминалом. В этом случае пользователь обязан убедиться, что терминал находится в А/Ц режиме. Это можно сделать с помощью подпрограммы ANMODE. В других случаях вызывать подпрограмму ANMODE не требуется, поскольку при необходимости она вызывается автоматически.

Форма обращения: CALL ANMODE

6.3. Вывод алфавитно-цифровых символов. Подпрограмма ANCHO

Вывод неуправляющих А/Ц символов осуществляется подпрограммой ANCHO. При необходимости включается А/Ц режим и, после окончания вывода символа, изменяется значение параметра положения луча в области состояния терминала. Если при выводе символа луч выходит за правый край, то автоматически генерируется переход на новую строку.

В качестве входного параметра предполагается применение 7-битового кода ASCII; код символа представляется целым числом. Подпрограмма ANCHO не проверяет правильность значения входного кода. Любой неиспользуемый числовой код вызовет занесение ошибочной информации о состоянии луча.

Подпрограмма ANCHO изменяет значения координат текущего положения луча соответственно размерам символов.
Форма обращения:
CALL ANCHO(ICHAR)
Входной параметр:
ICHAR — целое число, представляющее символ в 7-разрядном коде ASCII. Символ обязательно неуправляющий.

6.4. Вывод алфавитно-цифровых символов. Подпрограмма ANSTR

Подпрограмма ANSTR подобна подпрограмме ANCHO кроме того, что позволяет выводить алфавитно-цифровую строку вместо отдельного символа. Входными параметрами подпрограммы ANSTR являются число выводимых символов NCHAR и массив NADE десятичных эквивалентов символов в кодах ASCII, представляющих выводимую строку.
Форма обращения:
CALL ANSTR(NCHAR,NADE)
Входные параметры:
NCHAR — число выводимых символов
NADE — массив целых десятичных эквивалентов символов кода ASCII.

6.5. Обработка алфавитно-цифровых символов

6.5.1. Подпрограмма NEWLIN вызывает перевод строки и возврат каретки. Форма обращения:
CALL NEWLIN

6.5.2. Подпрограмма LINEF вызывает перевод строки. Форма обращения:
CALL LINEF

6.5.3. Подпрограмма CARTN вызывает возврат каретки. Форма обращения:
CALL CARTN

6.5.4. Подпрограмма HOME передвигает луч в левый верхний угол экрана. Форма обращения:
CALL HOME

6.5.5. Подпрограмма NEWPAG очищает экран терминала и возвращает луч в исходное положение HOME.
Форма обращения:
CALL NEWPAG

6.5.6. Подпрограмма BAKSP возвращает луч на одну позицию. Форма обращения:
CALL BAKSP

6.6. Использование экранного курсора

6.6.1. Подпрограмма SCURSR

Графический курсор можно использовать для прямого указания координат экрана. Обращение к подпрограмме SCURSR активизирует курсор, позволяя пользователю разместить его в требуемой позиции. Координаты курсора передаются в ЭВМ при нажатии клавиши символа. Этот символ вместе с координатами курсора возвращается как результат работы подпрограммы. Система управления терминалом компенсирует влияние курсора на положение луча.
Форма обращения:
CALL SCURSR(ICHAR,IX,IY)
Результаты:
ICHAR — символ клавиатуры в виде 7-разрядного кода ASCII.
IX — экранная X-координата курсора
IY — экранная Y-координата курсора.

Для управления курсором используется функциональная клавиатура. Клавиша с кодом «АР2-Н» переключает скорость передвижения курсора.

Следующий пример иллюстрирует использование экранного курсора. Подпрограмма ANMODE вызывается для печати координат курсора (рис. 10).

Пример:

```
      CALL INITT(30)
10    CALL SKURSR(ICHAR,IX,IY)
      CALL PNTABS(IX,IY)
      CALL ANMODE
      WRITE(6,20),IX,IY
20    FORMAT(1H+,$,I5,I4)
      IF(ICHAR.NE.83)GO TO 10
      CALL FINITT(0,0)
      CALL H
      END
```

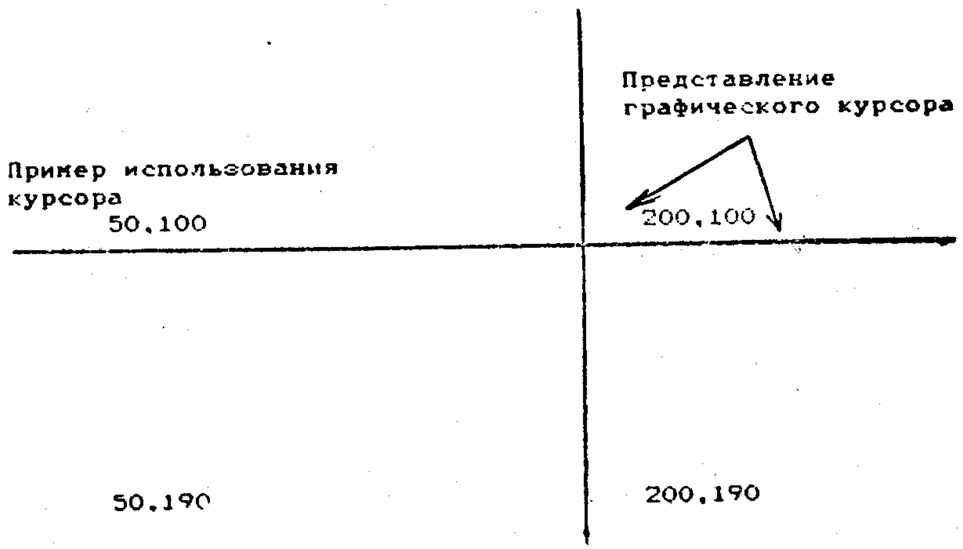


Рис. 10

6.6.2. Подпрограмма DCURSR выполняет те же функции, что и подпрограмма SCURSR. Форма обращения и аргументы такие же.

6.7. Использование виртуального курсора. Подпрограмма VCURSR

Часто удобнее определить виртуальное пространство, а не экранные координаты курсора. Подпрограмма VCURSR позволяет пользователю работать с графическим курсором. После размещения курсора, его экранные координаты передаются в ЭВМ при нажатии символа на клавиатуре. Подпрограмма VCURSR переводит входные данные в виртуальные координаты соответственно текущему описанию окна. Виртуальный курсор не влияет на положение луча. Преобразования, выполняемые подпрограммой VCURSR предполагают, что весь экран представляет виртуальное пространство, определенное текущим окном.

Форма обращения:

CALL VCURSR (ICHAR,X,Y)

Возвращаемые параметры:

ICHAR — 7-битный код символа, нажатого на клавиатуре
X — виртуальная X-координата графического курсора
Y — виртуальная Y-координата графического курсора

В следующем примере VCURSR позволяет чертить линии в режиме взаимодействия. При нажатии клавиши вычерчивается отрезок линии или передвигается луч от текущего положения к точке, указанной курсором (рис. 11).

Пример:

```
DIMENSION FRAME ( 8 )
DATA FRAME /1.,0., .,1.,0.,1.,0.,0./
CALL INITT (120)
C * определение минимумов и размеров окон
CALL SWINDO(50,390,50,250)
CALL VWINDO(0.,1.,0.,1.)
C * обрамление экранного окна
      CALL MOVEA(0.,0.)
      DO 100 I=1,8,2
100   CALL DRAWA(FRAME(I), FRAM(I+1))
C * повторяющийся вызов курсора
150   CALL ANMODE
      CALL VCURSR(IBR,X,Y)
C * нажата буква P точка
      IF (IBR.EQ.80) GO TO 200
C * нажата буква D черчение
      IF (IBR.EQ.68) GO TO 300
C * нажата буква M сдвиг
      IF (IBR.EQ.77) GO TO 400
C * нажата буква S конец
      IF (IBR.EQ.83) GO TO 500
      GO TO 150
200   CALL POINTA (X,Y)
      GO TO 150
300   CALL DRAWA (X,Y)
      GO TO 150
400   CALL MOVEA (X, Y)
      GO TO 150
500   CALL ANMODE
      CALL FINITT (0,67)
      END
```

Рис. 11

6.8. Область состояния терминала

Набор переменных, содержащихся в общем блоке и представляющих текущее состояние терминала, называется областью состояния терминала. Система управления терминалом позволяет пользователю сохранять текущее состояние терминала и позже в него возвращаться. Хотя здесь и не сохраняются выведенные данные, это средство позволяет пользователю прервать работу, перейти на другой участок программы для выполнения каких-либо вычислений или работы пользователя в интерактивном режиме и после этого вернуться к прерванному процессу.

Так как пользователь сам размещает сохраняемую информацию, то он может сохранить больше, чем один уровень состояний и восстанавливать любой из сохраняемых уровней состояния в любое время.

6.8.1. Подпрограмма SVSTAT

Текущее состояние терминала можно сохранить, передав подпрограмме сохранения статуса вещественный массив длиной 60 слов, в который можно записать область состояния терминала. Нельзя надежно сохранить и вернуть обратно статус пунктирных линий. Форма обращения:

```
CALL SVSTAT (RARRAY)
```

Входной параметр:

RARRAY — вещественный массив длиной 60 слов.

6.8.2. Подпрограмма RESTAT

Терминал можно в любое время настроить на любое ранее сохраненное состояние, передав в подпрограмму восстановления состояния вещественный массив длиной 60 слов, в котором хранится область состояния терминала.

Форма обращения:

```
CALL RESTAT (RARRAY)
```

Входной параметр:

RARRAY — вещественный массив длиной 60 слов, содержащий ранее сохраненное состояние терминала.

Пример: рисование фигуры, приведенной на рис. 12

```
      CALL INITT (30)
      DIMENSION IBOX(8), ITIME(8), B(60), T(60)
      DATE IBOX/200,0,0,200,—200,0,0,—200/
      DATE ITIME/80,0,—80,120,80,0,—80,—120/
      CALL MOVABS(400,300)
C * сохранение текущего состояния терминала
      CALL SVSTAT(B)
      CALL MOVABS(60,40)
      CALL SVSTAT(T)
      DO 100 N=1,7,2
      CALL DRWREL(ITIME(N), ITIME(N+1))
      CALL SVSTAT(T)
C * восстановление состояния терминала
      CALL RESTAT(B)
      CALL DRWREL(IBOX(N), IBOX(N+1))
      CALL SVSTAT(B)
100   CALL RESTAT(T)
      CALL FINITT(0,67)
      STOP
      END
```

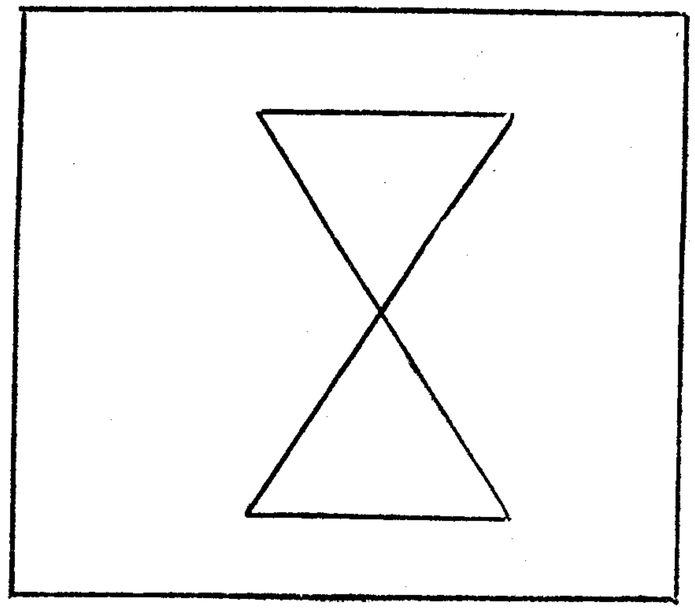


Рис. 12

6.9. Масштабирование и поворот

Для описания объектов и элементов чертежа применяются относительные векторы. Может потребоваться разместить описанные объекты в различных местах виртуального пространства, но размер и ориентация этих элементов не всегда должны быть одинаковыми. По этой причине относительные вектора автоматически масштабируются и поворачиваются подпрограммами относительных векторов в соответствии с коэффициентом масштабирования TRSCAL и коэффициентами поворота TRCOSF и TRSINF. Переменные TRSCAL, TRCOSF и TRSINF входят в состав переменных общей области параметров состояния терминала. Все параметры в

обращении к подпрограммам с относительными векторами указываются без масштабирования и поворота. Входные параметры определяют нормальные размеры и ориентацию для относительного вектора. Абсолютные вектора операциями масштабирования и поворота не подвергаются.

6.9.1. Установка масштабирования

Коэффициент масштабирования для относительного вектора TRSCAL может быть использован для изменения длины относительного вектора. Все относительные вектора масштабируются в соответствии с текущим значением коэффициента масштабирования. Например, если какой-либо объект записан в кодах относительных векторов и необходимо составить описание такого же объекта, но с размерами вдвое больше, то следует установить значение TRSCAL=2 и повторно обратиться к данному объекту.

Параметр относительного масштабирования может быть задан аналогично заданию значения любой переменной. Если масштабирование не требуется, то параметру TRSCAL должно быть приписано значение 1.0, которое устанавливается в начале работы подпрограммой INITT.

6.9.1.1. Масштабирование графического вывода. Подпрограмма RSCALE

Геометрическую фигуру, начерченную в относительных координатах, можно масштабировать любым виртуальным, относительным множителем соответственно описанию виртуального окна, то есть часть фигуры будет отсечена, если ее размеры превысят размеры виртуального окна.

Форма обращения:

CALL RSCALE (FACTOR)

Входной параметр:

FACTOR — масштабный множитель относительно начала чертежа.

6.9.2. Установка поворота

Направление относительных векторов можно изменить с помощью коэффициентов поворота TRCOSF и TRSINF. Коэффициент TRCOSF равен косинусу угла поворота, TRSINF— синусу угла поворота. Необходимо следить, чтобы сумма квадратов этих коэффициентов была равна 1.0, иначе появится искажение рисунка.

Все относительные векторы поворачиваются в соответствии с текущим значением коэффициента поворота. Если пользователь хочет получить изображение объекта, состоящее из относительных векторов, но ориентированное под углом, отличающимся от нормальной ориентации, то он должен задать

коэффициенты поворота равными косинусу и синусу угла поворота и обратиться к подпрограмме вычерчивания объекта.

Коэффициенты поворота относительного вектора могут быть заданы аналогично заданию обычных переменных. Если не требуется поворота относительных векторов, то значения коэффициентов поворота должны быть заданы: TRCOSF=1.0; TRSINF=0.0. Эти начальные значения устанавливаются подпрограммой INITT.

6.9.2.1. Поворот графического вывода. Подпрограмма RROTAT

Геометрическую фигуру, начерченную в относительных координатах, можно повернуть на любой угол относительно начальной позиции вывода. Форма обращения:

CALL RROTAT(DEG)

Входной параметр:

DEG — угол поворота в градусах относительно начальной позиции вывода.

В следующем примере вычерчивается треугольник, затем он масштабируется множителем 2 и поворачивается на 90 градусов, чтобы получить следующий треугольник (рис. 13).

```
      CALL INITT(30)
      CALL TRIANGL(200.,200.)
C *   двойное увеличение размера
      CALL RSCALE(2.)
C *   поворот на 90 градусов и черчение
      CALL TRIANGL(220.,140.)
      CALL FINITT(10,10)
      CALL H
      STOP
      END
      SUBROUTINE TRIANGL(X,Y)
      CALL MOVEA(X,Y)
      CALL MOVER(—50.,—50.)
      CALL DRAWR(100.,0.)
      CALL DRAWR(—50.,100.)
      CALL DRAWR(—50.,—100.)
      CALL POINTR(50.,50.)
      RETURN
      END
```

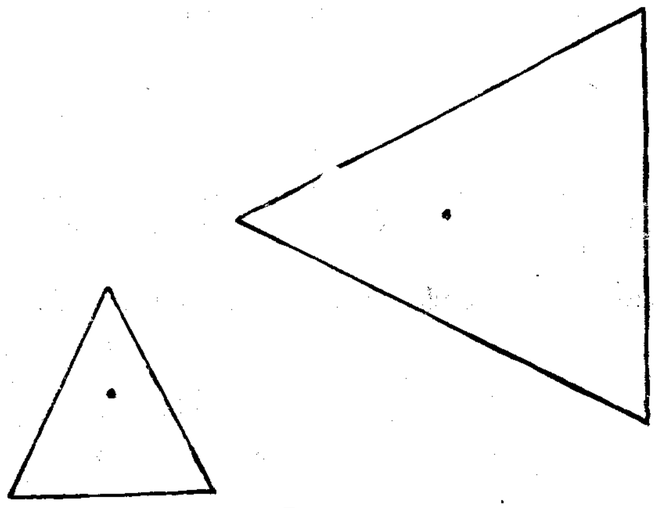


Рис. 13

6.10. Подпрограмма RESET выполняет те же действия, что и подпрограмма INITT, но не переходит на новую страницу. Форма обращения:
CALL RESET

6.11. Подпрограмма RECOVR приводит в соответствие режим работы аппаратуры терминала с данными, хранящимися в области состояний, терминала. Это удобно для вывода на терминал последующей информации, например, после применения оператора ФОРТРАНа WRITE, действие которого не контролируется системой управления терминалом. Форма обращения:
CALL RECOVR

6.12. Подпрограмма VECMOD вызывает перевод терминала в режим вывода векторов. Форма обращения:
CALL VECMOD

6.13. Подпрограмма DSHMOD устанавливает параметр пунктирной линии и вызывает переход в режим вычерчивания пунктирной линии. Форма обращения:
CALL DSHMOD(L)
Входной параметр: L — параметр пунктирной линии

6.14. Подпрограмма LVLCHT проверяет текущее состояние графического режима. Если в данный момент установлен режим непосредственного вывода, то подпрограмма выключает луч и переходит к режиму виртуальной графики. Форма обращения:
CALL LVLCHT

6.15. Различные служебные подпрограммы

6.15.1. Подпрограмма ERASE

Экран терминала можно очистить без изменения состояния и положения луча. Система запрещает генерацию дополнительного изображения, пока не закончено стирание. Форма обращения:
CALL ERASE

6.15.2. Подпрограмма BELL

Для привлечения внимания пользователя к определенному событию в любое время можно выдать звуковой сигнал. Часто продолжительный звуковой сигнал, генерируемый последовательностью обращений к подпрограмме звонка, используется для сигнала тревоги. «Звонок» может звучать в любом режиме, кроме режима GIN (графический ввод) и не влияет на состояние терминала. Форма обращения:
CALL BELL

6.15.3. Подпрограмма SEETW

Выдает текущие значения экранного окна. Форма обращения:
CALL SEETW (MINX, MAXX, MINY, MAXY)
Результаты:
MINX — минимальная горизонтальная экранная координата
MAXX — максимальная горизонтальная экранная координата
MINY — минимальная вертикальная экранная координата
MAXY — максимальная вертикальная экранная координата

6.15.4. Подпрограмма SEEDW

Выдает текущие значения пределов виртуального окна. Форма обращения:
CALL SEEDW (XMIN, XMAX, YMIN, YMAX)
Результаты:
XMIN — минимальная горизонтальная координата пользователя
XMAX — максимальная горизонтальная координата пользователя
YMIN — минимальная вертикальная координата пользователя
YMAX — максимальная вертикальная координата пользователя

6.15.5. Подпрограмма SEEREL

Выдает значения общих переменных, используемых подпрограммами масштабирования и поворота относительных векторов.
Форма обращения:
CALL SEEREL (RCOS, RSIN, SCALE)
Результаты:
RCOS — косинус угла поворота

RSIN — синус угла поворота
SCALE — множитель для масштабирования

6.15.6. Подпрограмма SEETRN

Подпрограмма выдает значения общих переменных, устанавливаемых процедурами определения окна и преобразования.

Форма обращения:
CALL SEETRN (XFAC, YFAC, KEY)
Результаты:
XFAC — масштабный множитель по оси X
YFA — масштабный множитель по оси Y
KEY — ключ преобразования:
— 1=линейное
KEY= — 2=логарифмическое
— 3=полярное

6.16. Перевод дюймов в экранные единицы. Функция KIN

Подпрограмма — функция KIN переводит дюймы в экранные единицы. Она определяет число единиц растра в (RI) дюймах.

Форма обращения:
переменная=KIN (RI)
Входной параметр:
RI — число дюймов
Результат:
KIN — число единиц растра в (RI) дюймах.
Пример: KIN является средством определения положения на экране, когда пользователь хочет работать с виртуальными единицами.

```
IX=KIN (1.4)
CALL DRWREL (IX, 0)
```

6.17. Перевод сантиметров в экранные единицы. Функция KCM

Подпрограмма-функция KCM переводит сантиметры в экранные единицы. Она определяет число единиц растра в (RC) сантиметрах. Форма обращения:
переменная=KCM (RC)
Входной параметр:
RC — число сантиметров
Результат:
KCM — число единиц растра в (RC) сантиметрах.
Пример: KCM является средством определения положения на экране, когда пользователь хочет работать с виртуальными единицами.

```
IX=KCM (3.5)
```


6.18. Измерение ширины символов. Функция LINWDT

Функция LINWDT определяет в единицах растра горизонтальный размер данного числа смежных символов.
Форма обращения:
переменная=LINWDT (NUMCHR)
Входной параметр:
NUMCHR — число смежных символов, для которых определяется ширина в единицах растра.
Результат:
LINWDT — ширина в единицах растра NUMCHR символов.

6.19. Измерение высоты строк. Функция LINHGT

Функция LINHGT выдает высоту данного числа строк в единицах растра.
Форма обращения:
переменная=LINHGT (NUMLIN)
Входной параметр:
NUMLIN — целое число строк, для которого определяется высота в единицах растра.
Результат:
LINHGT — высота (NUMLIN) строк в единицах растра.

6.20. Табуляция и задание границ (полей)

Система управления терминалом разрешает пользователю устанавливать и отменять табулятор и границы строки (поля) для облегчения задания формата. Установка табулятора и границ производится программно и полезна только при использовании подпрограмм вывода А/Ц информации. Все величины табулятора и границ задаются в экранных координатах.

Допускается вертикальная и горизонтальная табуляция, границы устанавливаются слева и справа. Горизонтальная и вертикальная табуляция ограничены 10-ю значениями.

6.20.1. Установка таблицы табулятора. Подпрограмма TTBLSZ

Значения вертикального и горизонтального табуляторов содержатся в двух целых массивах по десять слов каждый. Значения табулятора упорядочены по возрастанию экранных координат. Первое нулевое значение обозначает конец таблицы значений. Подпрограмма TTBLSZ устанавливает длину целого массива. Горизонтальный и вертикальный массивы должны быть равной длины. Форма обращения:
CALL TTBLSZ (ITBLSZ)
Входной параметр:
ITBLSZ — размер таблицы табулятора (горизонтальный и вертикальный), выраженный целым числом от 1 до 10.

6.20.2. Установка табулятора. Подпрограмма SETTAB

Подпрограмма SETTAB вставляет заданные значения табулятора в экранных координатах в заданную таблицу табулятора. Если таблица заполнена, то максимальные значения табулятора будут потеряны, при вставке меньших значений. При появлении такой ситуации устанавливается флаг общей ошибки KGNFLG. Хотя дублирующие установки не вставляются в таблицу, подпрограмма не проверяет правильность значения установки табулятора и не проверяет тип задаваемой таблицы для табулятора.

Форма обращения:

CALL SETTAB (ITAB, ITBTBL)

ITAB — значения табулятора в координатах X или Y

ITBTBL — горизонтальная или вертикальная таблица табулятора (имя массива).

6.20.3. Замена табулятора. Подпрограмма RSTTAB

При замене определенного значения табулятора его величина в экранных координатах должна быть введена в качестве значения параметра обращения к подпрограмме замены с указанием имени таблицы. Ненулевое значение, не совпадающее ни с одним значением, записанным в текущую таблицу табуляции, игнорируется. Замена всей таблицы выполняется путем задания нулевого значения первого параметра в обращении к подпрограмме замены. Форма обращения:

CALL RSTTAB (ITAB, ITBTBL)

Входные параметры:

ITAB — экранные X или Y координаты заменяемого табулятора. Если число равно 0, то все значения табулятора в таблице сбрасываются.

ITBTBL — горизонтальная или вертикальная таблица табулятора (имя массива).

6.20.4. Горизонтальная табуляция. Подпрограмма TABHOR

Обращение к подпрограмме TABHOR вызовет горизонтальное передвижение курсора с постоянным значением координаты Y в положение, определяемое первым ненулевым значением в таблице горизонтальной табуляции, которое больше текущей X — координаты курсора или позиции луча в экранных координатах. Если таблица горизонтального табулятора пуста, то никаких действий не производится. Если таблица не пуста и в ней нет значений, больших текущей экранной X-координаты курсора или положения луча, или если первое ненулевое значение больше экранной X-координаты и больше заданного значения положения правой границы, то генерируется переход на новую строку.

Форма обращения:

CALL TABHOR(ITBTBL)

Входной параметр:

ITBTBL — имя таблицы горизонтальной табуляции.

6.20.5. Вертикальная табуляция. Подпрограмма TABVER

Вертикальная табуляция вызывает вертикальное перемещение курсора с постоянным значением координаты X до позиции, определенной последним ненулевым значением в таблице вертикальной табуляции, которое меньше текущей Y-координаты курсора или положения луча. Если в таблице вертикальной табуляции нет ненулевого значения, которое меньше текущей координаты Y, то никаких действий не производится. Форма обращения:

CALL TABVER(ITBTBL)

Входной параметр:

ITBTBL — имя таблицы вертикальной табуляции.

В следующем примере показана установка и отмена табулятора, используя вывод символов с помощью подпрограммы ANCHO (рис. 14).

Пример:

```
      DIMENSION IHORZ(4), IVERT(4)
      CALL INITT(30)
      CALL TTBLSZ(4)
      CALL SETTAB(20, IHORZ)
      CALL SETTAB(50, IHORZ)
      CALL SETTAB(100, IHORZ)
      CALL SETTAB(150, IHORZ)
      CALL SETTAB(200, IVERT)
      CALL SETTAB(250, IVERT)
      CALL SETTAB(300, IVERT)
      CALL SETTAB(350, IVERT)
      DO 100 IVTAB=1, 4
      IF (IVTAB.EQ.3) CALL RSTTAB(250, IHORZ)
      IF (IVTAB.EQ.3) CALL SETTAB (270, IHORZ)
      CALL TABVER (IVERT)
      DO 50 IHTAB=1, 4
      CALL TABHOR(IHORZ)
      LTR=64+IHTAB+4*(IVTAB—1)
      CALL ANCHO(LTR)
50    CONTINUE
      CALL NEWLINE
100   CONTINUE
```

```
CALL FINITT(0, 0)
CALL H
END
```

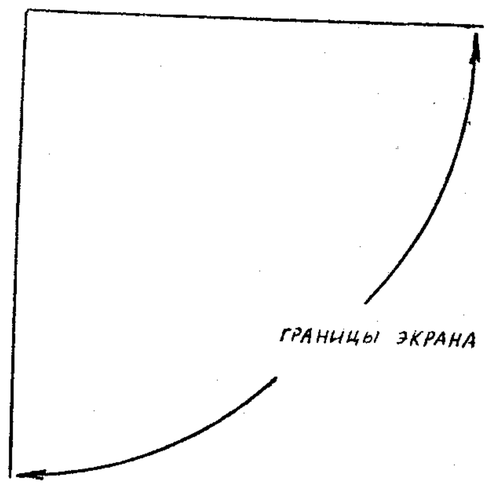


Рис. 14

6.20.6. Задание границ экрана. Подпрограмма SETMRG

Подпрограмма устанавливает значения левой и правой границ на экране, которые используются следующими программами:

— CARTN — возврата каретки
— HOME — возврата в исходное положение
— NEWPAG — вызова новой страницы

Левая граница определяет значение экранной координаты X, с которой начинается строка А/Ц символов. Значение левой границы хранится в виде переменной KLMRGN в области состояния терминала. Начальное значение левой границы устанавливается равным 0 подпрограммой INITT.

Правая граница определяет самое правое положение на экране, где может размещаться выводимый символ. Попытка осуществить вывод символа за пределами правой границы вызовет переход к новой строке. Правая граница представляет собой значение X координаты в экранной системе и хранится в виде переменной KRMRGN в области состояния терминала.

Форма обращения:

CALL SETMRG (MLEFT, MRIGHT)

Входные параметры:

MLEFT — значение экранной координаты, с которой начи-

ается вывод строки А/Ц символов. Эта величина должна ыть больше 0 и меньше максимальной экранной координаты (399) или значения правой границы;

MRIGHT — значение экранной координаты, в которой кончается вывод строки А/Ц символов. Ее величина всегда должна быть больше MLEFT, но меньше максимальной экранной координаты (399).

6.21. Вспомогательные подпрограммы управления выводом графической информации

6.21.1. Очистка текущего экранного окна. Подпрограмма CLEAR

Иногда пользователю необходимо стереть не весь экран, а только текущее экранное окно. Для этой цели в пакете предусмотрена подпрограмма CLEAR. Форма обращения:

CALL CLEAR(I)

Тип стирания зависит от значения параметра I:

0 — стирание;
1 — установка.

6.21.2. Установка типа рисуемой линии. Подпрограмма LNSTYL

Кроме вышеописанных подпрограмм отрисовки пунктирных линий у пользователя есть возможность установить тип рисуемой линии для всех последующих вызовов подпрограмм отрисовки линий, как в виртуальном пространстве, так и в экранном. Это производит подпрограмма LNSTYL. Форма обращения.

CALL LNSTYL(I)

Тип линии зависит от значения параметра I и не изменяется до следующего вызова подпрограммы.

0 — сплошная линия;
1 — чередование пробелов и штрихов в одну растровую единицу;
2 — чередование пробелов и штрихов в две растровых единицы;
3 — чередование пробелов и штрихов в три растровых единицы;
4 — чередование пробелов и штрихов в четыре растровых единицы;
5, 6, 7 — различные типы штрих-пунктирных линий.

После инициализации пакет настроен на рисование сплошными линиями.

6.21.3. Установка типа рисования. Подпрограмма SETMOD

Иногда существует необходимость стереть не все экранное окно, а только некоторые фигуры. Для этой цели в пакет

введена подпрограмма управления режимом рисования.
Форма обращения:
CALL SETMOD(I)
Режим рисования зависит от параметра I:
0 — рисование стиранием;
1 — рисование установкой в единицу.
После инициализации, пакет настроен на рисование установкой.

7. ПРЕОБРАЗОВАНИЕ СИСТЕМ КООРДИНАТ

Подпрограммы преобразования позволяют пользователю определить одну из трех координатных систем: линейную, логарифмическую или полярную. По умолчанию используется линейное преобразование (LINTRN), которое работает впредь до изменения системы преобразования. Обращение к подпрограмме LINTRN снова возвращает систему к линейному преобразованию в пределах заданного окна.

Логарифмическое преобразование, устанавливаемое подпрограммой LOGTRN позволяет пользователю отображать данные в логарифмическом масштабе как по отдельным осям X или Y, так и по обеим одновременно.

Полярное преобразование, устанавливаемое подпрограммой POLTRN, позволяет пользователю определять координаты в единицах радиуса-вектора и градусах.

Каждое преобразование выполняется автоматически перед вычерчиванием вектора; оно остается в силе, пока не будет вызвано другое преобразование или до реинициализации системы.

Пример: преобразования, изменяющие систему координат, в которой определены данные.

POLTRN	радус, угол	10,0	10,90	10,180
LINTRN	X, Y	10,0	0,10	—10,0

На рис. 15, полученном с помощью приведенной ниже программы, вычерчена сетка, вычисляемая подпрограммой пользователя GRID (приведена в тексте примера). Эта сетка показана в пяти различных системах координат. Сетка (1) линейная. Сетки (2), (3), (4) показывают три различных типа логарифмического преобразования; соответственно: (X — логарифм, Y — линейна), (X — линейна, Y — логарифм) и (X — логарифм, Y — логарифм). Сетка (5) демонстрирует результат обращения к подпрограмме полярного преобразования POLTRN. Все 5 сеток используют одни и те же виртуальные данные. Разница между ними получается из-за типа преобразования действующего в момент формирования изображения.

Пример:

```
      CALL INITT(30)
C *   определение окна данных
      CALL DWINDO(10., 100., 10., 100.)
C *   описание экранного окна (1)
      CALL TWINDO(0, 120, 180, 270)
C *   черчение сетки, показывающей линейное преобразование
      CALL GRID
C *   определение экранного окна (2)
      CALL TWINDO(0, 120, 0, 90)
      CALL LOGRTN(1)
C *   черчение сетки, показывающей преобразование X-логарифм,
C *   Y — линейна
      CALL GRID
C *   определение экранного окна (3)
      CALL TWINDO(270, 390, 180, 270)
      CALL LOGTRN(2)
C *   черчение сетки, показывающей преобразование X — линейна,
C *   Y — логарифм
      CALL GRID
C *   определение экранного окна(4)
      CALL TWINDO(270, 390, 0, 90)
      CALL LOGTRN(3)
C *   черчение сетки, показывающей преобразование X — логарифм,
C *   Y — логарифм
      CALL GRID
C *   определение экранного окна (5)
      CALL TWINDO(140, 260, 90, 180)
      CALL POLTRN(10., 100., 0.)
C *   черчение сетки, показывающей полярное преобразование
      CALL GRID
      CALL FINITT(0, 0)
      CALL H
      END
```

```
C * черчение сетки линиями от 10 до 100 с интервалом 10
      SUBROUTINE GRID
      DMIN=10.
      DMAX=100.
      X=DMIN
C * черчение линий сетки вдоль оси X
      DO 100 I=1, 10
      CALL MOVEA(X, DMIN)
      CALL DRAWSA(X, DMAX)
100   X=X+10.
      Y=10.
C * черчение линий сетки вдоль оси Y
      DO 200 J=1, 10
      CALL MOVEA(DMIN, Y)
      CALL DRAWSA(DMAX, Y)
200   Y=Y+10.
      RETURN
      END
```

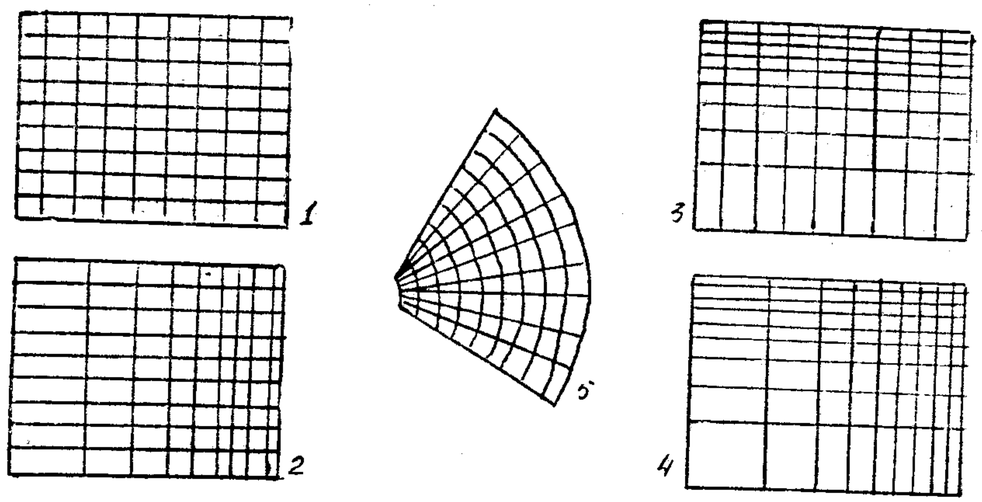


Рис. 15

7.1. Линейное преобразование. Подпрограмма LINTRN

Подпрограмма восстанавливает режим линейного масштабирования после работы в логарифмической или полярной системе координат. В начале работы в системе управления терминалом действует линейная система координат. Форма обращения:

CALL LINTRN

7.2. Логарифмическое преобразование. Подпрограмма LOGTRN

Подпрограмма определяет либо ось X либо ось Y, либо обе, как логарифмические шкалы в пределах заданных размеров окна пользователя. Форма логарифмического масштабирования указывается параметром KEY. Форма обращения:

CALL LOGTRN(KEY)

Входной параметр:

KEY = — 1 — ось X логарифмическая
— ось Y линейная
— 2 — ось X линейная
— ось Y логарифмическая
— 3 — ось X логарифмическая
— ось Y логарифмическая

7.3. Полярное преобразование. Подпрограмма POLTRN

Подпрограмма POLTRN позволяет пользователю описать его виртуальные графические данные в полярных координатах. Полярные координаты определяются радиусом и углом. Угол представляется в градусах, положительное направление — против часовой стрелки от горизонтальной прямой, вправо от начала координат. Аргументы подпрограммы POLTRN определяют форму экранного окна, в котором вычерчиваются виртуальные данные. Виртуальное окно масштабируется и преобразовывается для заполнения области экрана между аргументами ANGMIN и ANGMAX. Третий аргумент, RSUPRS, вычитается из виртуального радиуса. Если ANGMAX и ANGMIN не равны максимуму и минимуму (YMAX и YMIN) окна данных (DWINDO), или если RSUPRS не равно 0, то форма графика, вычерченного по виртуальным данным в полярных координатах, будет искажена (см. рис. 17 и рис. 18). Пользователь может приспособить такие искажения для выделения любых желаемых форм вычерчивания графиков.

Начало полярных координат автоматически устанавливается так, чтобы разместить наибольшую возможную область вывода внутри экранного окна пользователя. Форма обращения:

CALL POLTRN(ANGMIN, ANGMAX, RSUPRS)

Входные параметры:

ANGMIN — минимальный угол от горизонтали, от которого появляется изображение на экране;

ANGMAX — максимальный угол от горизонтали, до которого появляется изображение на экране;
RSUPRS — коэффициент подавления радиуса.

7.4. Вычерчивание сегментов с использованием полярного преобразования. Подпрограммы DRAWSA и DRAWSR

В подпрограмме GRID для вычерчивания сетки (5) на рис. 15 использовалось обращение к подпрограмме DRAWSA вместо подпрограммы DRAWA. Подпрограмма DRAWSA аналогична подпрограмме DRAWA, за исключением того, что она позволяет пользователю вычерчивать сегменты кривых линий, необходимых при работе в полярных координатах. Используемая при работе в полярной системе координат подпрограмма DRAWSR аналогична подпрограмме DRAWR.
Форма обращения:
CALL DRAWSA(X, Y)
Входные параметры:
X, Y — виртуальные координаты точки, до которой вычерчивается сегмент линии.
Форма обращения:
CALL DRAWSR(X, Y)
Входные параметры:
X, Y — виртуальные координаты, задаваемые относительно текущей позиции луча.

7.5. Черчение пунктирных сегментов линий в полярной системе координат. Подпрограммы DASHSA и DASHSR.

Подпрограммы DASHSA и DASHSR работают аналогично подпрограммам DASHA и DASHR соответственно. Они применяются при работе в полярной системе координат. Форма обращения:
CALL DASHSA(X, Y, L)
Входные параметры:
X, Y — виртуальные координаты точки, до которой должен быть проведен сегмент пунктирной линии;
L — тип пунктирной линии.
Форма обращения: CALL DASHSR(X, Y, L)
Входные параметры:
X, Y — значения виртуальных координат точки, в которую должен быть проведен сегмент пунктирной линии относительно текущего положения луча;
L — тип пунктирной линии.

Ниже приведена сводная таблица подпрограмм черчения системы управления графическим терминалом.

Таблица 4

Экранные координаты. Целые значения			Виртуальные координаты. Вещественные значения	
1			2	
Действие	Абсолютные	Относительные	Абсолютные	Относительные
Перемещение	MOVABS	MOVREL	MOVEA	MOVER
Черчение	DRWABS	DRWREL	DRAWA	DRAWR
Точка	PNTABS	PNTREL	POINTA	POINTR
Пунктирная линия	DSHABS	DSHREL	DASHA	DASHR
Черчение сегмента	нет	нет	DRAWSA	DRAWSR
Черчение пунктирного сегмента	нет	нет	DASHSA	DASHSR

7.6. Применение полярных преобразований

Используя сетку, заданную в полярных координатах (рис. 15) исследуем возможности, предоставляемые программой POLTRN для вычерчивания различных форм графиков. Пример 1 демонстрирует проведение сегментов пунктирной линии, соединяющей 30 точек, имеющих различные значения длины радиуса в пределах от 90. до 100., заданных через каждые 3 градуса в пределах от ANGMIN=10 градусов до

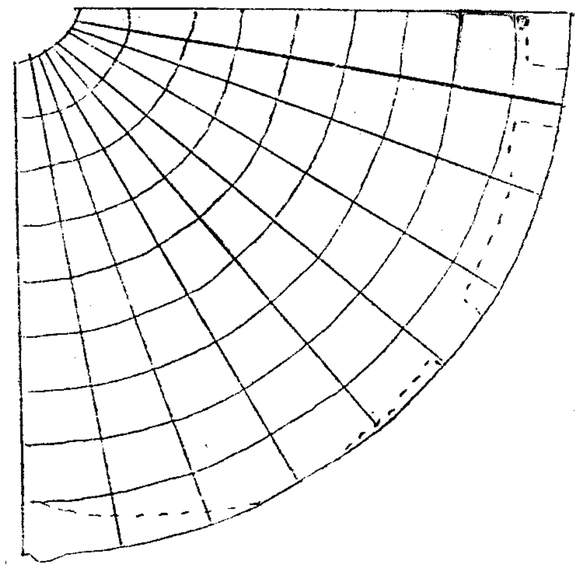


Рис. 16

ANGMAX=100 градусов (рис. 16). Коэффициент подавления радиуса равен 0.

Пример 1:

```
      DIMENSION RDATA(30)
      DATA RDATA/90.3, 92.4, 94.5, 95.2, 96.1, 96.9, 98.2, 98.7,
     *99.1, 99.4, 99.7, 100.0, 99.5, 99.0, 98.5, 98.0, 97.5,
     *97.0, 96.5, 96.0, 95.5, 95.0, 94.0, 93.0, 92.0, 93.0,
     *97.0, 94.3, 91.0, 92.8/
      CALL INITT(30)
C *   определение окна на экране терминала
      CALL TWINDO(0, 390, 0, 270)
C *   задание границ окна с изменением
C *   радиуса от 10 до 100
      CALL DWINDO(10., 100., 10., 100.)
C *   задание полярной системы координат с
C *   изменением угла в пределах окна
C *   от 10 до 100 градусов и с нулевым
C *   коэффициентом подавления радиуса
      CALL POLTRN(10., 100., 0.)
C *   вычерчивание сетки в пределах окна
      CALL GRID
C *   вычерчивание графика по заданным значениям
C *   радиуса через каждые 3 градуса в пределах
C *   от 10 до 100 градусов
      CALL MOVEA(RDATA(1),10.)
      DO 10 I=1,30
      DEGREE=10+I*3
10    CALL DASHSA(RDATA(I),DEGREE,12)
      CALL FINITT(0,0)
      CALL H
      END
      SUBROUTINE GRID
      DMIN=10.
      DMAX=100.
      X=DMIN
      DO 100 I=1,10
      CALL MOVEA(X.DMIN)
      CALL DRAWSA(X,DMAX)
100   X=X+10.
      Y=10.
      DO 200 J=1,10
      CALL MOVEA(DMIN,Y)
      CALL DRAWSA(DMAX,Y)
200   Y=Y+10.
```



```
      RETURN
      END
```

В примере 2 показывается, как будет выглядеть тот же график, если размеры виртуального окна (по радиусу) заданы в пределах от 90 до 100 градусов, коэффициент подавления радиуса опять будет равным нулю. Следует отметить, что при выводе отсекаются все линии сетки, имеющие значение радиуса меньше 90 градусов (рис. 17).

Пример 2:

```
      SUBROUTINE GRID
      DMIN=10.
      DMAX=100.
      X=DMIN
      DO 100 I=1,10
      CALL MOVEA(X,DMIN)
      CALL DRAWSA(X,DMAX)
100   X=X+10.
      Y=10.
      DO 200 J=1,10
      CALL MOVEA(DMIN,Y)
      CALL DRAWSA(DMAX,Y)
200   Y=Y+10.
      RETURN
      END
      DIMENSION RDATA(30)
      DATA RDATA/90.3,92.4,94.5,95.2,96.1,96.9,98.2,98.7,
     *99.1,99.4,99.7,100.0,99.5,99.0,98.5,98.0,97.5,
     *97.0,96.5,96.0,95.5,95.0,94.0,93.0,92.0,93.0,
     *97.0,94.3,91.0,92.8/
      CALL INITT (30)
C *   определение окна на экране терминала
      CALL TWINDO(0,390,0,270)
C *   задание пределов изменения радиуса от 90 до 100
      CALL DWINDO(90.,100.,10.,100.)
C *   определение полярного окна с изменением угла
C *   от 10 до 100 градусов и коэффициентом
C *   подавления радиуса 90
      CALL POLTRN(10.,100.,90.)
C *   вычерчивание сетки в пределах окна
      CALL GRID
C *   вычерчивание графика по данным  с приращением по
      углу
C *   через 3 градуса в пределах от 10 до 100 градусов
      CALL MOVEA(RDATA(1),10.)
      DO 10 I=1,30
      DEGREE=10+I*3
10    CALL DASHSA(RDATA(I),DEGREE,12)
```

```
CALL FINITT(0,0)
CALL H
END
```

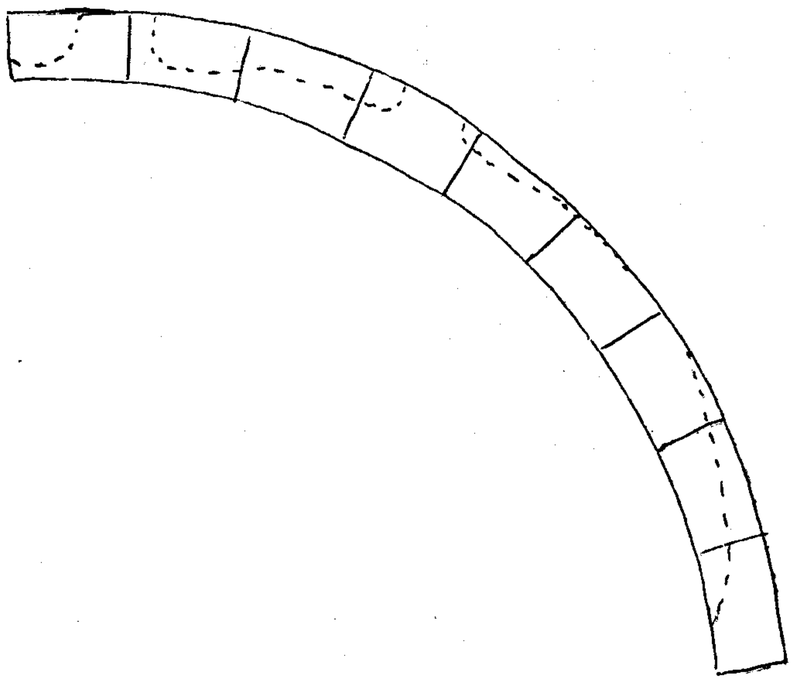


Рис. 17

В примере 3, на основе тех же точек графика, как и в примере 2, показывается изменение формы графика полярного окна с изменением угла от 10 до 100 (рис. 18)

```
      SUBROUTINE GRID
      DMIN=10.
      DMAX=100.
      X=DMIN
      DO 100 I=1,10
      CALL MOVEA(X,DMIN)
      CALL DRAWSA(X,DMAX)
100   X=X+10.
      Y=10.
      DO 200 J=1,10
      CALL MOVEA(DMIN,Y)
      CALL DRAWSA(DMAX,Y)
```



```
200   Y=Y+10.
      RETURN
      END
      DIMENSION RDATA(30)
      DATA RDATA/90.3,92.4,94.5,95.2,96.1,96.9,98.2,98.7,
     *99.1,99.4,99.7,100.0,99.5,99.0,98.5,98.0,97.5,
     *97.0,96.5,96.0,95.5,95.0,94.0,93.0,92.0,93.0,
     *97.0,94.3,91.0,92.8/
      CALL INITT (30)
C * определение окна на экране терминала
      CALL TWINDO(0,390,0,270)
C * задание пределов изменения радиуса от 90 до 100
      CALL DWINDO(90.,100.,10.,100.)
C * определение полярного окна  с изменением угла от 10
    до 100
C * градусов и коэффициентом подавления радиуса 90
      CALL POLTRN(10.,100.,90.)
C * вычерчивание сетки в пределах окна
      CALL GRID
C * вычерчивание графика по данным  с приращением по
    углу
C * через 3 градуса в пределах от 10 до 100 градусов
      CALL MOVEA(RDATA(1),10.)
      DO 10 I=1,30
      DEGREE=10+I*3
10    CALL DASHSA(RDATA(I),DEGREE,12)
      CALL FINITT(0,0)
      CALL H
      END
```

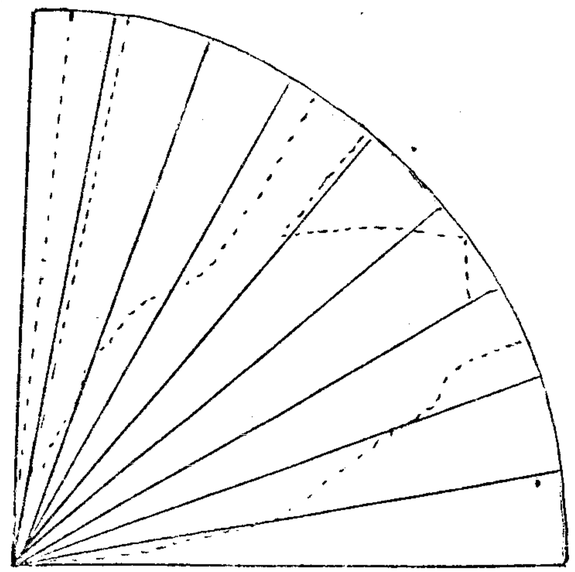


Рис. 18

В примере 4 используются все данные примера 2, только изменены пределы полярного окна от ANGMIN=0 до ANGMAX=180. Коэффициент подавления радиуса оставлен без изменения равным 90 (рис. 19).

Пример 4:

```
      SUBROUTINE GRID
      DMIN=10.
      DMAX=100.
      X=DMIN
      DO 100 I=1,10
      CALL MOVEA(X,DMIN)
      CALL DRAWSA(X,DMAX)
100   X=X+10.
      Y=10.
      DO 200 J=1,10
      CALL MOVEA(DMIN,Y)
      CALL DRAWSA(DMAX,Y)
200   Y=Y+10.
      RETURN
      END
      DIMENSION RDATA(30)
      DATA RDATA/90.3,92.4,94.5,95.2,96.1,96.9,98.2,98.7,
     *99.1,99.4,99.7,100.0,99.5,99.0,98.5,98.0,97.5,
     *97.0,96.5,96.0,95.5,95.0,94.0,93.0,92.0,93.0,
     *97.0,94.3,91.0,92.8/
      CALL INITT(30)
C * определение окна на экране терминала
      CALL TWINDO(0,390,0,270)
C * задание пределов изменения радиуса от 90 до 100
      CALL DWINDO(90.,100.,10.,100.)
C * определение полярного окна с изменением угла
C * от 10 до 100 градусов и коэффициентом подавления
C * радиуса 90
      CALL POLTRN(0.,180.,90.)
C * вычерчивание сетки в пределах окна
      CALL GRID
C * вычерчивание графика по данным с приращением по
      углу
C * через 3 градуса в пределах от 10 до 100 градусов
      CALL MOVEA(RDATA(1),10.)
      DO 10 I=1,30
      DEGREE=10+I*3
10    CALL DASHSA(RDATA(I),DEGREE,12)
      CALL FINITT(0,0)
      CALL H
      END
```

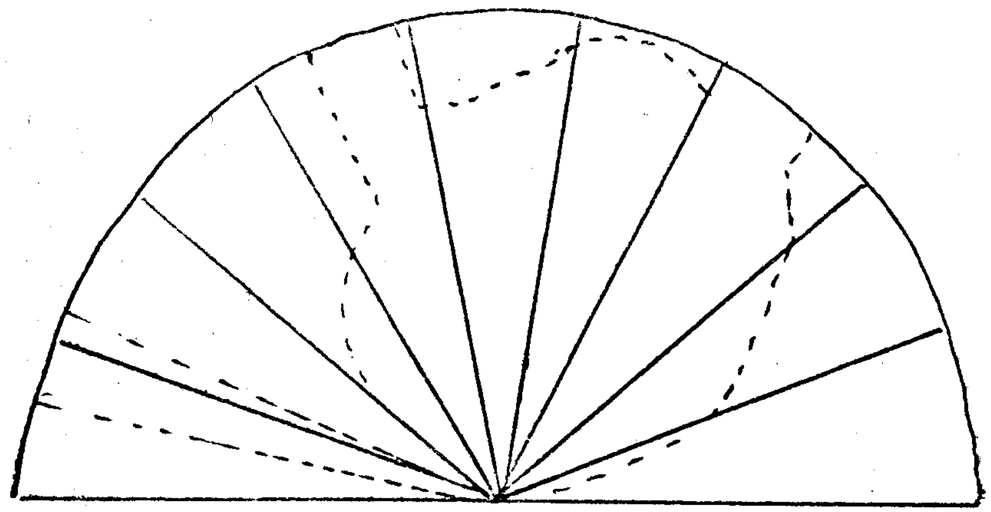



Рис. 19

ПРИМЕЧАНИЕ: Размерности массива RDATA и начальные значения его элементов в примерах 2,3,4 перед обращением к подпрограмме INITT такие же как и в примере 1.

8. ПОДПРОГРАММЫ ВВОДА/ВЫВОДА

Программа пользователя с помощью подпрограмм системы управления терминалом может выполнять операции ввода/вывода с тремя видами информации: графической, алфавитно-цифровой и управляющей для терминала и периферийных устройств. Все операции по выводу осуществляются через подпрограмму TOUTST, а входные сигналы передаются через подпрограмму TINSTR. В операциях по вводу/выводу графической и управляющей информации эти подпрограммы используются непосредственно (наряду с их двойниками TOUTPT и TINPUT для одиночного символа). Для ввода/вывода алфавитно-цифровой информации может быть использован более гибкий набор подпрограмм, описанный ниже.

Для операций вывода система управления терминалом преобразует все символы, которые должны пересылаться в терминал в форму десятичного эквивалента кодов ASCII (ADE) и упаковывает их в выходной буфер. Когда буфер полностью заполнен или система (или пользователь) обращается к подпрограмме TSEND, этот буфер вызывается, преобразуется в коды используемой системы передачи данных и передается в терминал.

При вводе данных система управления терминалом воспринимает системно-зависимые коды, поступающие от терминала, преобразует их в десятичный эквивалент и направляет в качестве входных данных в запросившую подпрограмму, в случае необходимости производится также преобразование в алфавитно-цифровой формат.

С точки зрения пользовательской программы ввод алфавитно-цифровой информации более эффективно может быть реализован на основе использования прямых методов, таких как операторы READ и WRITE языка ФОРТРАН. Однако, вывод через систему обеспечивает коррекцию положения луча на экране терминала (за исключением специально отмеченных случаев) и позволяет размещать символы точно в любом заданном месте экрана дисплея. Ввод через систему обеспечивает правильную форматизацию данных для последующего вывода или внутреннего использования. На ответственность пользователя возлагается обращение к подпрограмме ANMODE для вызова содержимого выходного буфера до выполнения операций ввода/вывода языка ФОРТРАН.

При вводе и выводе используется три формата описания информации:

1 ADE (десятичный эквивалент кодов ASCII)
2 A1 (одно слово с размещением в нем одного символа)
3 AM (размещение M алфавитно-цифровых символов в одном слове;
число M равно максимальному количеству символов, помещаемых в одном машинном слове), M=2

8.1. Вывод

8.1.1. Подпрограмма TOUTST осуществляет вывод на экран терминала массива десятичных эквивалентов символов. Эта подпрограмма не изменяет значений координат положения луча на экране, хранящихся внутри системы управления терминалом и не переводит терминал в алфавитно-цифровой режим работы. Подпрограмма TOUTST должна использоваться только при выводе управляющих символов, которые при ином способе передачи не будут обрабатываться системой управления терминалом.

Форма обращения:
CALL TOUTST(NCHAR,IARRAY)
Входные параметры:
NCHAR — длина массива IARRAY, т. е. число выводимых символов;
IARRAY — имя массива, в котором хранятся десятичные эквиваленты передаваемых символов.

8.1.2. Подпрограмма TOUTPT выводит десятичный эквивалент одного символа. Эта подпрограмма не изменяет значений координат положения луча на экране, хранящихся внутри системы управления терминалом, и не переводит терминал в алфавитно-цифровой режим работы. Подпрограмма TOUTPT должна использоваться только при выводе управляющих символов, которые при ином способе передачи не будут обрабатываться системой управления терминалом. Форма обращения:
CALL TOUTPT (ICHAR)
Входной параметр:
ICHAR — десятичный эквивалент передаваемого символа.

8.1.3. Подпрограмма ANCHO производит вывод десятичного эквивалента одного символа. Одновременно эта подпрограмма переводит терминал в алфавитно-цифровой режим, выводит символ и изменяет текущее положение луча на экране.

Форма обращения:
CALL ANCHO (ICHAR)
Входной параметр:
ICHAR — десятичный эквивалент неуправляющего символа, подлежащего выводу.

8.1.4. Подпрограмма ANSTR выполняет те же функции, как и подпрограмма ANCHO за исключением того, что она применяется для вывода массива неуправляющих символов. Подпрограмма ANSTR также переводит терминал в алфавитно-цифровой режим и изменяет значение текущего положения луча, хранящееся внутри системы управления терминалом.
Форма обращения:
CALL ANSTR(NCHAR,IARRAY)
Входные параметры:
NCHAR — число выводимых символов;
IARRAY — имя массива, содержащего целочисленные десятичные эквиваленты передаваемых символов кода ASCII.

8.1.5. Подпрограмма A1OUT производит вывод массива символов, записанных в формате A1 языка ФОРТРАН. Эта подпрограмма переводит терминал в алфавитно-цифровой режим и заменяет текущее положение луча в системе управления терминалом по окончании вывода.
Форма обращения:
CALL A1OUT (NCHAR,IARRAY)
Входные параметры:
NCHAR — длина массива IARRAY, равная числу выводимых символов;

IARRAY — массив подлежащих выводу символов, записанных в формате A1 языка ФОРТРАН.

8.1.6. Подпрограмма AOUTST выводит массив символов в формате AM. В этом формате число M представляет количество символов, записываемых в одно машинное слово. Эта подпрограмма также редактирует значения координат положения луча на экране терминала, хранящиеся в системе управления терминалом.

Форма обращения:

CALL AOUTST (NCHAR, IARRAY)

Входные параметры:

NCHAR — число символов, подлежащих выводу, должно быть в M раз больше длины массива IARRAY (количества слов в массиве)

IARRAY — массив подлежащих выводу символов, записанных в формате AM. Если длина массива IARRAY короче, чем определено параметром NCHAR, то недостающие символы дополняются пробелами.

8.2. Ввод

8.2.1. Подпрограмма TINSTR воспринимает входные сигналы от терминала и помещает в массив десятичных эквивалентов ADE. Эти символы хранятся в форме, пригодной для вывода с помощью подпрограммы ANCHO или ANSTR.

Форма обращения:

CALL TINSTR(LEN, IARRAY)

Входной параметр:

LEN — количество ожидаемых символов. Если число полученных символов меньше, чем задано параметром LEN, то массив IARRAY дополняется пробелами. Если принято больше символов, то избыток будет введен при следующем обращении к подпрограмме TINSTR.

Возвращаемые данные:

IARRAY — массив десятичных эквивалентов, в который записываются принятые символы.

8.2.2. Подпрограмма TINPUT воспринимает от терминала десятичный эквивалент одного символа. Форма обращения:

CALL TINPUT (ICHAR)

Возвращаемый параметр:

ICHAR — десятичный эквивалент кода символа, полученного от терминала. Поскольку подпрограмма TINPUT обращается к подпрограмме TINSTR, то нулевая запись (появляющаяся при нажатии на клавишу возврата каретки) становится пробелом, но при вводе более одного символа все они хранятся

для последующей выборки при любом обращении к подпрограмме TINSTR.

8.2.3. Подпрограмма A1IN воспринимает от терминала массив символов в целочисленном формате A1. Массив записывается в таком виде, который пригоден для непосредственного вывода с помощью подпрограммы A1OUT.

Форма обращения:

CALL A1IN(NCHAR, IARRAY)

Входной параметр:

NCHAR — число ожидаемых символов от терминала. Поскольку подпрограмма A1IN обращается к подпрограмме TINSTR, то при получении менее NCHAR символов массив IARRAY дополняется пробелами; если получено больше символов, чем ожидалось, то оставшиеся символы запоминаются и вызываются при последующих обращениях к подпрограмме TINSTR.

Возвращаемые данные:

IARRAY — массив, в котором размещаются принятые символы в формате A1.

8.2.4. Подпрограмма AINST принимает с терминала массив символов в формате AM. Этот массив затем может быть выведен подпрограммой AOUTST. Форма обращения:

CALL AINST(NCHAR, IARRAY)

Входной параметр:

NCHAR — число ожидаемых символов. Поскольку подпрограмма AINST обращается к подпрограмме TINSTR, то при получении меньшего количества символов, чем число NCHAR, массив IARRAY дополняется пробелами, если получено больше символов, то оставшиеся символы записываются в память для вызова при любом следующем обращении к подпрограмме TINSTR.

Возвращаемые данные:

IARRAY — массив, в котором размещаются символы в формате AM.

8.3. Подпрограммы обслуживания ввода/вывода

Последующие подпрограммы помогают пользователю системы управления терминалом организовать ввод и вывод данных. Эти подпрограммы следует использовать очень осторожно и в целом ряде случаев к ним не стоит обращаться.

8.3.1. Проверка свободного места во входном буфере. Подпрограмма-функция LEFTIO

Подпрограмма-функция LEFTIO определяет оставшееся число символов во входном буфере или свободное место (в символах), имеющееся в выходном буфере. В тех случаях,

когда количество вводимой информации переменное, пользователь, например, может пожелать уточнить, сколько еще символов он может ввести для получения заданного количества. Форма обращения:

K=LEFTIO(IBUFF)

Входной параметр:

IBUFF — показывает, какой буфер должен проверяться.

K — число символов, оставшееся незаполненным в буфере, заданном параметром IBUFF.

8.3.2. Определение положения графического луча на экране. Подпрограмма SEELOC

Данная подпрограмма обеспечивает пользователю средство для определения последнего положения графического луча на экране, если перед этим был реализован вывод информации помимо системы управления терминалом (например, оператор READ или WRITE языка ФОРТРАН или путем обращения к подпрограммам TOUTSTR или TOUTPT). Таким образом пользователь имеет возможность самостоятельно изменить значение положения луча.

Форма обращения:

CALL SEELOC (IX, IY)

Возвращаемые данные:

IX — экранная координата положения луча по оси X;

IY — экранная координата положения луча по оси Y.

9. РАСШИРЕННОЕ ИСПОЛЬЗОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ТЕРМИНАЛОМ

Комбинируя простые подпрограммы системы управления терминалом, можно создать очень сложные прикладные программы. В приводимом примере показан способ использования графического курсора совместно с программами вычерчивания и перемещения для интерактивного формирования чертежа сложной электронной схемы (рис. 21, 22).

Основная программа обращается к подпрограмме виртуального курсора. Пользователь может установить курсор в любой точке экрана.

Нажатие различных функциональных кнопок на терминале вызовет перемещение луча в эту точку или вычерчивание отрезка прямой линии в данную точку или переход к подпрограмме, которая формирует строку символов, начинающуюся с данной позиции.


```
      Пример :
C  *  программа интерактивного черчения схем
      DATA IDRAW/68/,IMOVE/77/,IERASE/69/,IQUIT/81/,IHCOPY/72/
      DATA IRESIS/82/,ICAP/67/, ITRANS/84/,IGRND/71/
      CALL INITT(30)
C  *  установка экранного окна на терминале
      CALL TWINDO(0,390,0,270)
C  *  установка окна в виртуальном пространстве
      CALL DWINDO(0.,500.,0.,375.)
      CALL MOVEA(0.,0.)
C  *  обращение к графическому курсору
100   XFROM=XTO
      YFROM=YTO
105   CALL UCURSR(KEY,XTO,YTO)
      IF(KEY.NE.IDRAW)GO TO 110
      CALL DRAWA(XTO,YTO)
      GO TO 100

  110     IF (KEY.NE.IMOVE)GO TO 120
          CALL MOVEA(XTO,YTO)
          GO TO 100
  120     IF(KEY.NE IERASE)GO TO 130
          CALL ERASE
          GO TO 105
  130     IF(KEY.NE.IQUIT)GO TO 140
          CALL FINITT(0.0)
          CALL H
  140     IF(KEY.NE.IHCOPY)GO TO 150
          CALL HDCOPY
          GO TO 105
C  *  определение поворота элемента
  150     RANGLE=ATAN2(YTO—YFROM,XTO—XFTROM)*57.2957795131
          CALL PROTATE(RANGLE)
          IF(KEY.NE.IRESIS)GO TO 160
          CALL RESIST
          CALL DRAWA(XTO,YTO)
          GO TO 100
  160     IF(KEY.NE.ICAP)GO TO 170
          CALL CAP
          CALL DRAWA(XTO,YTO)
          GO TO 100
  170     IF(KEY.NE.ITRANS)GO TO 180
          CALL TRANS
          CALL MOVEA(XFROM,YFROM)
C  *  луч в начальной точке транзистора
          GO TO 105
  180     IF(KEY.NE.IGRND)GO TO 100
          CALL GROUND
          CALL MOVEA(XFROM,YFROM)
C  *  луч в начальной точке символа «земля»
          GO TO 105
          END
```

Подпрограммы черчения четырех различных элементов (рис. 20):

Символ	Обозначение
сопротивление	R
конденсатор	C
транзистор	T
земля	G

```
C * подпрограмма черчения символа сопротивления
      SUBROUTINE RESIST
      CALL DRAWR(10.,0.)
      CALL DRAWR(0.,7.)
      CALL DRAWR(36.,0.)
      CALL DRAWR(0.,—14.)
      CALL DRAWR(—36.,0.)
      CALL DRAWR(0.,7.)
      CALL MOVER(36.,0.)
      CALL DRAWR(10.,0.)
      RETURN
      END
C * подпрограмма черчения символа конденсатор
      SUBROUTINE CAP
      CALL DRAWR(10.,0.)
      CALL MOVER(0.,20.)
      CALL DRAWR(0.,—40.)
      CALL MOVER(10.,0.)
      CALL DRAWR(0.,40.)
      CALL MOVER(0.,—20.)
      CALL DRAWR(10.,0.)
      RETURN
      END
C * подпрограмма черчения символа транзистор
      SUBROUTINE TRANS
      CALL DRAWR(20.,0.)
      CALL DRAWR(0.,20.)
      CALL DRAWR(2.,0.)
      CALL DRAWR(0.,—40.)
      CALL DRAWR(—2.,0.)
      CALL DRAWR(0.,20.)
      CALL MOVER(2.,10.)
      CALL DRAWR(20.,20.)
      CALL MOVER(—20.,—40.)
      CALL DRAWR(15.,—15.)
      CALL DRAWR(2.,2.)
```



```
      CALL DRAWR(3.,—7.)
      CALL DRAWR(—9.,3.)
      CALL DRAWR(2.,2.)
      RETURN
      END
```

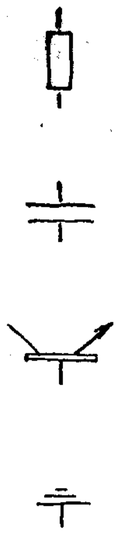


Рис. 20

Примеры схем, которые могут быть получены с помощью приведенных здесь программ приведены на рис.21 и рис.22.

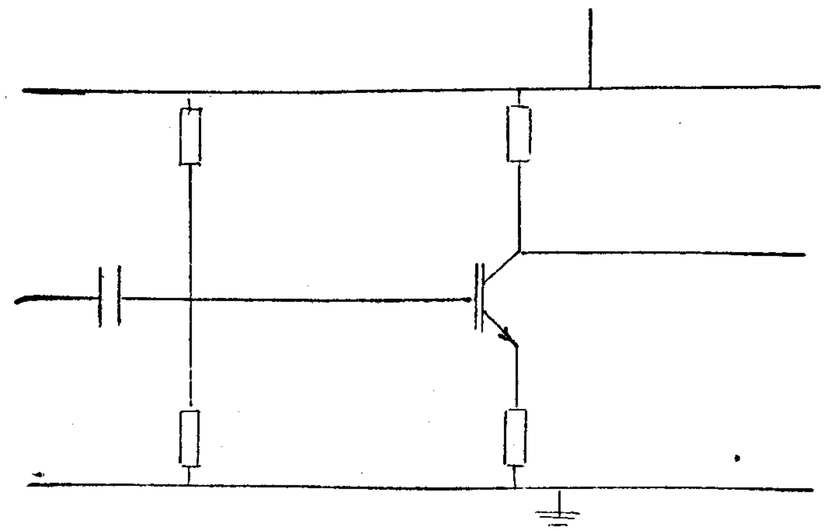


Рис. 21

```
C * подпрограмма черчения символа земли
      SUBROUTINE GROUND
      CALL DRAWR(10.,0.)
      CALL MOVER(0.,—16.)
      CALL DRAWR(0.,32.)
      CALL MOVER(2.,—28.)
      CALL MOVER(2.,—20.)
      CALL DRAWR(0.,16.)
      RETURN
      END
```

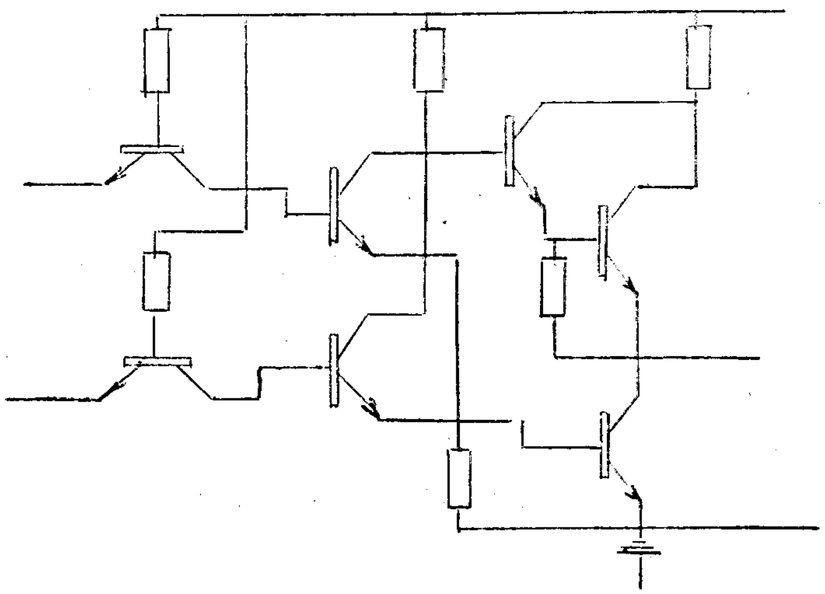


Рис. 22

10. ОБЩИЕ (ГЛОБАЛЬНЫЕ) ПЕРЕМЕННЫЕ ОБЛАСТИ СОСТОЯНИЯ ТЕРМИНАЛА

10.1. Область общих переменных

В системе управления терминалом определена область общих (глобальных) переменных, отображающая текущее состояние терминала и режимы работы, установленные пользователем. Область состояния терминала описана в каждом случае применения системы управления терминалом как блок общей памяти, легко доступный для всех подпрограмм пользователя.

Некоторые параметры, содержащиеся в области состояний терминала, такие как ширина символа (KHORS) и его высота (LVERSZ), оказываются очень полезные для всех программ независимо от терминала. Другие переменные, например, коэффициент масштабирования относительного вектора (TRSCAL) и его коэффициенты поворота (TRCOSF, TRSINF), или параметры границ строки (KLMRGN, KRMRGN), должны быть доступны для тех подпрограмм, которым эти параметры необходимы. Квалифицированный программист при использовании системы управления терминалом может легко воспользоваться информацией, хранящейся в области состояний, и при грамотном применении это позволит значительно увеличить возможности его программ.

Не все переменные из области состояний терминала используются одновременно в любых прикладных программах.

10.2. Порядок описания COMMON области

Область состояния терминала определена ниже как меченый COMMON (общий) блок, обычный для программ на языке ФОРТРАН IV. Общему блоку присвоено имя TKTRNX для всех применений. Порядок переменных в общем блоке должен совпадать с порядком, описанным ниже.

Все переменные области состояний терминала будут представлять целые или действительные числа в соответствии с обычными соглашениями языка ФОРТРАН о типе переменной, соответствующей заданному имени.

```
      Общая область системы управления терминалом
COMMON /TKTRNX/ TMINVX,TMINVY,TMAXVX,TMAXVY,TREALX,TREALY,
1 TIMAGX,TIMAGY,TRCOSF,TRSINF,TRSCAL,TRFACX,TRFACY,
2 TRPAR1,TRPAR2,TRPAR3,TRPAR4,R5,TRPAR6,KMOFLG(2),
3 KGNMOD,KPADV,KACHAR,KOBLEN,KTRAIL,KLEVEL,KPAD2,
4 KBAUDR,KGNFLG,KGRAFL,KHOMEY,KKMODE,KHORSZ,KVERSZ,KTBLSZ,
5 KSIZEF,KLMRGN,KRMRGN,KFACTR,KTERM,KLINE,KZAXIS,KBEAMX,KBEAMY,
6 KMOVEF,KPCHAR(5),KDASHT,KNINSX,KMINSY,KMAXSX,KMAXSY,KEYCON,
7 KINLFT,KOTLFT,KUNIT
```

Название переменных, используемых в системе управления терминалом:

скорость передачи KBAUDR

— число символов, которое может быть передано на терминал за одну секунду. Для непосредственно подключенных терминалов значение этой переменной равно нулю.

Флажок графического режима KGRAFL·

— установка флажка означает выполнение пользователем операций в режиме виртуальной графики. При сбросе флажка предполагается, что работа выполняется в режиме непосредственного вывода графической информации.

Начальное значение Y KHOMEY

— положение точки по оси Y в экранных координатах, с которой начинается вывод строки символов на экран терминала.
Режим KKMODE
— переменная состояния, показывающая текущий режим работы терминала:
0 — алфавитно-цифровой,
1 — вычерчивание векторов,
2 — вывод точек,
3 — вывод в приращениях,
4 — пунктир.

Параметры для алфавитно-цифровых символов
Перечисленные ниже символы используются только при обработке алфавитно-цифровой информации.
Горизонтальный размер символов KHORSZ
— число единиц растра в экранных координатах, на которое луч смещается по вертикали при аппаратной генерации перехода на следующую строку символов.
Флажок размера KSIZEF
— при установке флажка разрешается вычерчивание символов, увеличенных по размеру вдвое.
Левая граница KLMRGN
— левая граница строки в экранных координатах по оси X.
Правая граница KRMGRN
— правая граница строки в экранных координатах по оси X.
Размер таблицы табулятора KTBLSZ
— число слов в каждой из таблиц табулятора.

Таблица горизонтальной табуляции KHORSZ
— целочисленный массив из 10 слов, содержащий текущие значения табулятора. Записанные значения должны быть выражены в единицах экранных координат и расположены в порядке возрастания. Первое нулевое значение используется для индикации конца записей в таблице.
Параметры изображения на экране
Перечисленные здесь параметры используются на базовом уровне вывода графической информации.
Координаты луча по оси X KBEAMX
— экранная координата X текущего положения луча.
Изменяется каждый раз при перемещении луча в соответствии с полученной командой.
Координаты луча по оси Y KBEAMY
— экранная координата Y текущего положения луча.
Изменяется каждый раз при перемещении луча в соответствии с полученной командой.
Флажок перемещения KMOVEF

— флажок устанавливается для индикации получения команды на вычерчивание невидимого вектора в режиме вычерчивания векторов.
Предыдущие символы KPCHAR
— целочисленный массив из 4 слов, содержащий символы вычерчивания, которые определяют последний вычерченный вектор или точку.
Параметр пунктирной линии KDASHT
— определяет длины видимых и невидимых частей пунктирной линии.
Параметры виртуальной графики
Эти параметры используются совместно с непосредственным графическим выводом.
Нижняя граница X экранного окна KMINSX
— минимальное значение экранной координаты X для текущего окна на экране.
Нижняя граница Y экранного окна KMINSY
— минимальное значение экранной координаты Y для текущего окна на экране.
Верхняя граница X экранного окна KMAXSX
— максимальное значение экранной координаты X для текущего окна на экране.
Верхняя граница Y экранного окна KMAXSY
— максимальное значение экранной координаты Y для текущего окна на экране.
Виртуальная графика
KEYCON
— задает ключ преобразований при работе с виртуальной графикой.
KGNFLG
— задает флажок общей ошибки.
KGRAFL
— задает флажок графического режима.
KINLFT
— задает смещение символов влево во входном буфере.
KLINE
— задает тип вектора для программ графики.
KMOFLG
— может быть использовано для расширения пакета графических программ.
KOTLFT
— задает смещение символов влево в выходном буфере.
KPAD2
— задает размер заполнения.

KTERM
— задает тип терминала.
KUNIT
— задает формат выходного буфера.
TRFACX, TRFACY, TRPAR1, ..., TRPAR6
— задают параметры преобразований в виртуальной графике.
Нижняя граница X виртуального окна TMINVX
— минимальное значение координаты X для текущего виртуального окна.
Нижняя граница Y виртуального окна TMINVY
— минимальное значение координаты Y для текущего виртуального окна.
Верхняя граница X виртуального окна TMAXVX
— максимальное значение координаты X для текущего виртуального окна.
Верхняя граница Y виртуального окна TMAXVY
— максимальное значение координаты Y для текущего виртуального окна.
Действительная X — координата луча TREALX
— виртуальная координата X текущего действительного положения луча.
Действительная Y — координата луча TREALY
— виртуальная координата Y текущего действительного положения луча.
Воображаемая X — координата луча TIMAGX
— виртуальная координата X текущего воображаемого положения луча.
Воображаемая Y — координата луча TIMAGY
— виртуальная координата Y текущего воображаемого положения луча.
Косинус — коэффициент относительного вектора TRCOSF
— значение косинуса, используемое для поворота относительного вектора в виртуальном пространстве.
Синус — коэффициент относительного вектора TRSINF
— значение синуса, используемое для поворота относительного вектора в виртуальном пространстве.
Коэффициент масштабирования относительного вектора TRSCAL
— величина, используемая для масштабирования относительных векторов в виртуальном пространстве (только для применений, использующих арифметику с плавающей точкой).

11. ИНСТРУКЦИЯ ПО РАБОТЕ С ПАКЕТОМ

11.1. Работа с библиотекой

Пакет графических подпрограмм поставляется в виде объектной библиотеки PLTLIB.OBJ.

11.2. Трансляция, компоновка и запуск графических программ

Программы, использующие графические вызовы транслируются как и любые другие программы, написанные на языке ФОРТРАН.

Пример:

```
.R FORTRAN
*MX1:F,LP:=MX1:F
```

Оттранслированная программа должна быть скомпонована с библиотекой графических подпрограмм.

Пример:

```
.R LINK
*MX1:F=MX1:F, MX1:PLTLIB
```

Запуск программы осуществляется стандартным способом.

Пример:

```
RUN MX1:F
```

12. СЛОВАРЬ ТЕРМИНОВ СИСТЕМЫ УПРАВЛЕНИЯ ТЕРМИНАЛОМ

ABSOLUTE VECTOR
— абсолютный вектор — направленный отрезок прямой линии из начальной точки в заданную конечную точку. При выводе изображения на экран начальная точка определяется текущим положением луча, конечная точка определяется в абсолютных экранных координатах. В виртуальной графике начальная точка совпадает с положением виртуального луча, конечная точка задается в абсолютных экранных координатах.
ADE
— десятичный эквивалент кода ASCII. Целочисленное представление кодов символов ASCII (см. таблицу кодов ASCII).
ALFANUMERIC CURSOR
— алфавитно-цифровой курсор — прямоугольный перемещаемый маркер (не записываемый в память), указывающий позицию, в которой будет размещен следующий выводной символ.
ALFANUMERIC MODE
— алфавитно-цифровой режим работы терминала, при кото-

ром коды ASCII на выводе интерпретируются как символы, подлежащие отображению.

A/N — А/Ц
— Сокращение для слова «алфавитно-цифровой».

ASCII
— обозначение американского стандартного кода для обмена информацией: стандартный код, состоящий из 7-разрядных элементов.

BUFFERING
— буферизация — сохранение входных и выходных данных (в системе управления терминалом для этой цели имеется массив для 72 символов). Этот массив передается или распечатывается при полном заполнении или по специальной команде распечатки.

CLIPPING
— отсечение — такое изменение векторов в виртуальной графике, при котором части векторов, выходящие за пределы виртуального окна, не выводятся на экран дисплея.

COORDINATE
— координата — упорядоченная пара (X, Y) чисел, однозначно описывающая точку либо в экранном, либо в виртуальном пространстве. Упорядоченная пара чисел в нормальной системе координат (декартовой системе) описывает точку по ее расстоянию от начала координат (0,0) вдоль осей X и Y соответственно.

CURSOR
— курсор — перемещаемый маркер, используемый для целей указания положения на экране.

DEFOCUS MODE
— режим расфокусировки — вызывает утолщение линий на экране дисплея.

DRAW
— вычерчивание — команда для дисплея, вызывающая появление подсвеченного вектора на экране.

ERASE
— стирание — процедура удаления изображения с экрана терминала.

GRAFIC CURSOR
— курсор из пересекающихся линий на экране для определения положения вводимой точки.

HARDCOPY
— твердая копия на бумаге изображения, имеющегося на экране терминала, выполняется с помощью отдельного специального устройства.

HOME POSITION
— исходное положение — позиция в верхнем левом углу экрана, с которой нормально начинается печать символов на данной странице.

INPUT
— ввод данных, посылаемых с терминала на ЭВМ. Также данные, подготовленные для программы.

KEY BOARD
— клавиатура — часть терминала, которая дает возможность пользователю вводить А/Ц данные в ЭВМ.

LEFT MARGIN
— левый край — X — координата на экране, которая задает начальную точку строки при выводе алфавитно-цифровых символов.

MOVE
— перемещение — команда для дисплея, вызывающая перемещение луча без включения подсветки.

NEW LINE
— новая строка — операция, вызывающая перевод алфавитно-цифрового курсора к левому краю и на одну строку вниз.

NEW PAGE
— новая страница — операция, вызывающая стирание изображения на экране и переводящая алфавитно-цифровой курсор в исходное положение.

ORIGIN
— начало координат — точка, имеющая значение координат (0, 0). Начало координат на экране размещается в нижнем левом углу. Виртуальное пространство, по определению, имеет свое начало координат в центре.

POINT
— точка — команда для дисплея, вызывающая подсветку точки.

RASTER UNIT
— единица растра — расстояние между двумя соседними точками на экране; основная характеристика разрешающей способности терминала.

REFRESH
— регенерация — возобновление изображения на экране дисплея. Если терминал работает в режиме без заполнения, то изображение на экране элт должно постоянно возобновляться пользовательской программой, чтобы оно оставалось видимым.

RELATIVE VECTOR
— относительный вектор — средство описания абсолютного

вектора, который вычерчивается относительно текущего положения луча.

RIGHT MARGIN

— правый край — X-координата, задающая правую границу вывода алфавитно-цифровой строки. Любая попытка записать символ на экране за пределами правого края при использовании подпрограммы вывода А/Ц информации вызовет переход к новой строке.

SCREEN

— экран — часть терминала, на котором появляется изображение, формируемое на ЭВМ.

SCREEN COORDINATES

— экранные координаты — набор точек, которые образуют экран. Эти точки представляют двухмерное пространство в диапазоне от (0, 0) до (1023023) включительно. Экранные координаты должны быть всегда целые числа.

SCREEN WINDOW

— экранное окно — часть экрана, на которую путем масштабирования и переноса отображается виртуальное окно.

SOFTWARE

— программное обеспечение — программы и подпрограммы управляющие работой ЭВМ.

STORAGE BEAM

— записывающий луч — пучок электронов, который управляет внешними сигналами при записи символов и векторов на экране терминала.

STORAGE TUBE

— запоминающая элт — электронно-лучевая трубка (элт), на экране которой однажды записанное изображение сохраняется неопределенно долго, пока не будет стерто специально.

TERMINAL

— терминал — внешнее устройство ЭВМ.

TERMINAL STATUS

— состояние терминала — текущее состояние терминала.

TERMINAL STATUS AREA

— область состояний терминала — набор общих переменных, которые описывают текущее состояние терминала.

TIMESHARING

— разделение времени — режим использования ЭВМ для обслуживания нескольких пользователей фактически одновременно. Связь с ЭВМ, работающей в режиме разделения времени, обычно осуществляется через интерактивный терминал.

TRANSFORMATION

— преобразование — взаимосвязь между виртуальными и

экранными окнами. Преобразование может включать масштабирование, перенос и/или изменение координатных систем.

USER COORDINATES

— координаты пользователя — система координат, в которой единицы измерения устанавливаются пользователем (см. виртуальные координаты).

VECTOR

— вектор — отрезок прямой линии. Вектор может быть подсвеченным (видимым) или неподсвеченным (невидимым). Первый генерируется подпрограммой DRAW, второй — подпрограммой MOVE.

VIRTUAL COORDINATES

— виртуальный курсор — позволяет пользователю определить координаты в виртуальном пространстве с помощью графического курсора

VIRTUAL SPASE

— область структурированного изображения, определяемая пользователем, независимая от терминала.

WHITE—THROUGH MODE

— смешанный режим — позволяет выводить на экран терминала регенерируемое изображение одновременно с сохраняемым изображением.

MODULA-2. ОПИСАНИЕ ЯЗЫКА

1. ВВЕДЕНИЕ

Язык MODULA-2 возник из потребности в языках высокого уровня, достаточно эффективных и реализуемых на миникомпьютерах с целью использования в системном программировании. Его предшественниками являются языки PASCAL и MODULA. От последнего он получил имя, важное понятие модуля и систематизированный современный синтаксис. От языка PASCAL — большинство остальных свойств. Это, в частности, структуры данных, такие, как: массивы, записи, множества и указатели. Структурные предложения содержат операторы IF, CASE, REPEAT, WHILE, FOR и WITH. Их синтаксис таков, что каждый оператор заканчивается символом END. Большое влияние на разработку языка MODULA-2 оказал языка MESA. Широкие возможности его изощренной системы оказали существенное влияние на реализацию языка MODULA-2.

Язык, по-существу, машинно-независим, за исключением ограничений, связанных с длиной слова. Машинная независимость противоречит основному принципу языков системного программирования, согласно которому должна существовать возможность выражения на этом языке любых операций, присущих данной машине. Эта проблема разрешается с помощью понятия модуля. Машинно-зависимые особенности программ могут быть введены в специальные модули. Тогда их использование может быть эффективным и, в то же время, будет ограничено и изолировано. При этом могут быть ослаблены ограничения, например, на совместимость типов данных.

Обычно в системных языках имеются возможности описывать процедуры ввода-вывода с преобразованиями, подпрограммы обработки файлов, процедуры управлением памятью, планированием процессов и т. д. В языке MODULA-2 эти свойства не включены в сам язык, но могут появляться, как модули низкого уровня, используемые затем другими

программами. Такой набор стандартных модулей является существенной частью реализации языка MODULA-2.

Понятие процессов и их синхронизация через сигналы, так как это сделано в языке MODULA, заменено понятием более низкого уровня — сопрограммы. Однако, возможно описать некий стандартный модуль, который реализует такие процессы и сигналы. Отсутствие процессов и сигналов в языке позволяет программисту подобрать свой алгоритм планирования, связанный с задачей и описать соответствующий модуль. В простых, но часто встречаемых случаях, планировщик может вообще отсутствовать (например, когда параллельные процессы протекают только в драйверах устройств).

Современные языки для системного программирования должны также облегчать создание больших программ, возможно разрабатываемых несколькими людьми. Модули, написанные отдельными программистами, должны иметь хорошо специфицированные интерфейсы, которые могут быть описаны независимо от реализации данного модуля. В языке MODULA-2 это достигается разделением на модули определений (DEFINITION MODULE) и модули реализаций (IMPLEMENTATION MODULE). В модуле определений задается список объектов, доступных извне для соответствующего модуля реализации, причем только тех объектов, которые относятся к интерфейсу.

В р. 15 описана конкретная реализация языка MODULA-2. Эта программная система состоит из пятипроходного компилятора, компановщика объектных модулей и загрузчика. Компилятор проверяет не только правильность синтаксиса данного модуля, но и межмодульные связи, используя для этого информацию, полученную при компиляции других модулей.

В р. 16 описан набор широко используемых модулей. Приведены соответствующие модули определений, дополненные описаниями функций экспортируемых процедур.

2. ОПИСАНИЕ СИНТАКСИСА

Для описания синтаксиса языка используется расширенный формализм БЭКУСА—НАУРА. Синтаксические понятия (нетерминальные символы) выражаются обычными фразами, поясняющими их интуитивный смысл. Символы языка (терминальные символы) являются либо словами, либо строками, заключенными в кавычки. Каждое синтаксическое правило имеет вид: S=E

где S — синтаксическое понятие, а E — синтаксическое выражение.
Выражение E — имеет вид:

T1\T2\... \TN (N>0)

\

где TI является термами для E. S может принимать значение какого-либо одного терма из набора термов E. Каждый терм T имеет вид:

F1 F2 ... FN (N>0)

где FI являются факторами для T. Каждый фактор F — либо символ (терминальный или нетерминальный), либо записан в виде [E], обозначающем или синтаксическое выражение E, или пустое предложение. Если же фактор записан в виде <E>, то это обозначает либо пустое предложение, либо E, EE, EEE,.... Обычные скобки используются для группировки термов и факторов. Такой формализм (расширенная форма (БЭКУСА—НАУРА) позволяет описать даже собственный синтаксис.

Пример:

```
$синтаксис     =<предложение>.
$предложение   =нетермсим «=» выражение «.».
$выражение     =терм <«\» терм>.
$терм          =фактор <« » фактор>.
$фактор        =термсим\нетермсим\ «(«выражение»)».
$              \«[«выражение»]»\ «<«выражение»>».
```

3. БАЗОВЫЕ ПОНЯТИЯ ЯЗЫКА

Языком называется бесконечный набор предложений (программ). Предложения составлены в соответствии с синтаксисом. Каждое предложение (программа) это конечный набор символов из конечного словаря. В словарь языка MODULA-2 входят: идентификаторы, числа, строки, операторы и разделители. Они называются лексическими символами или лексемами и, в действительности, состоят из последовательности знаков алфавита (отметим разницу между символами и знаками). Представление символов посредством знаков зависит от имеющегося набора знаков. В данной работе используется набор КОИ-7 вместе со следующими правилами:

1. Идентификаторы — последовательность букв и цифр. Первый знак должен быть буквой.

```
$Идентификатор=буква <буква\цифра>
```


Примеры: X SCAN MODULA-2 русский процесс

ПРИМЕЧАНИЕ. Использование в идентификаторах букв ч, ш, щ, э, ю как строчных, так и прописных запрещено.

2. Числа — целые или вещественные без знака. Целые — последовательность цифр. Если число заканчивается буквой B, то оно рассматривается как восьмеричное число, если буквой H, то шестнадцатиричное, если буквой C, то это — представление знака в восьмеричном коде и тип CHAR (см. 6.1). Целое I в диапазоне 0<I<MAXINT может рассматриваться как INTEGER или CARDINAL. Если диапазон I MAXINT< <I<MAXCARD, то I должно быть типом CARDINAL. Для 16-битовых ЭВМ MAXINT=32767, а MAXCARD=65535. Вещественные числа должны содержать десятичную точку. Кроме того, они могут содержать десятичный множитель. Тип вещественного числа — REAL.

```
$Число                 = целое\вещественное.
$целое                 = цифра<цифра>\восьмеричная_цифра<вось-
                         меричная_цифра>
$                        ("C")\цифра<шестнадцат_цифра     >"H".
$вещественное          = цифра<цифра> "." <цифра>[показатель_
                         степени ].
$показатель_степени    = "E"["+"\"-"]цифра<цифра>.
$шестнадцат_цифра      = цифра\"A"\"B"\"C"\"D"\"E"\"F".
$цифра                 = восьмеричная_цифра\"8"\"9".
$восьмеричная_цифра    = "0"\"1"\"2"\"3"\"4"\"5"\"6"\"7".
Примеры: 1980 37648  78CH  0A3CH 33C  12.3  45,67E-8  10.E4
```

3. Строки — это последовательность знаков, заключенных в кавычки. Как двойные кавычки, так и апострофы могут использоваться для этой цели. Однако, открывающие и закрывающие кавычки должны быть одинаковы и они не должны встречаться внутри текста. Строки не могут продолжаться на следующую строку, т. е. разбиваться символом <ВК>.

```
$Строка=«'»<знак>«'»\'«'<знак>'»'.
```

Строка из одного знака имеет тип CHAR, из N знаков имеет тип ARRAY [0..N—1] OF CHAR

Примеры:

«MODULA-2» и «DON'T WORRY» 'пароль «Москва»'

4. Операторы и разделители — это специальные знаки, пары знаков или зарезервированные слова, приведенные ниже. Эти зарезервированные слова пишутся заглавными буквами, и их нельзя употреблять в качестве идентификаторов. Символ < > и # синонимы, также как & и AND.

+	=	AND	FOR	QUALIFIED
-	#	ARRAY	FROM	RECORD
*	<	BEGIN	IF	REPEAT
/	>	BY	IMPLEMENTATION	RETURN
:=	<>	CASE	IMPORT	SET
&	<=	CONST	IN	THEN

>=		DEFINITION	LOOP	TO
.	..	DIV	MOD	TYPE
,	:	DO	MODULE	INTIL
(	)	ELSE	NOT	VAR
[	]	END	OF	WHILE
{	}	EXIT	POINTER	WITH
^	|	EXPORT	PROCEDURE	

5. Пробелы внутри лексем не допускаются (за исключением строк). Многократные пробелы и переводы строки рассматриваются как один разделитель.
6. Комментарии — это произвольный набор знаков, заключенных в специальные кавычки «(*» и «*)». Они могут быть внесены между любыми двумя лексемами в программе. Комментарии могут быть вложенными (при этом первая кавычка «*)» не обязательно конец комментария).

4. ОПИСАНИЕ ИДЕНТИФИКАТОРОВ

Любой идентификатор, встречающийся в программе, должен быть описан, если только он не является стандартным. Последние рассматриваются, как предварительно описанные и они постоянны во всех частях программы. Такие идентификаторы называются глобальными. Описания также служат для обозначения таких свойств объектов, как: является ли объект константой, типом, переменной, процедурой или модулем.

Идентификаторы можно использовать лишь внутри сферы действия описаний. Обычно, эта область распространяется на целый блок (описание процедуры или модуля).

Эти общие правила дополнены следующими:
1. Если идентификатор X, описанный в D1, используется в описании D2, то описание D1 должно текстуально предшествовать описанию D2. Исключение составляет описание ссылочного типа. Тип «T», используемый в этом описании (POINTER TO T), может быть описан позже в этом же блоке.
2. Если идентификатор, определенный в модуле M1, описан как экспортный, область его действия распространяется и на блок, содержащий M1. Если M1 — единица компиляции (см.

15), действие описания распространяется лишь на те единицы, которые импортируют M1.
3. Идентификаторы полей в описании записи не должны совпадать только внутри описания записи. Вне его эти идентификаторы могут быть использованы для других целей.

Идентификаторы могут квалифицироваться. Для этого они дополняются впереди префиксом, обозначающим модуль (см. 11), в котором описаны эти идентификаторы. Префикс и идентификаторы разделяются точкой.

$Квалидент=идентификатор <«.» идентификатор>.

Ниже приведены стандартные идентификаторы и разделы, в которых они описаны:

ABS	10.2	INGL	10.2
ADR	10.2	INTEGER	6.1
ASH	10.2	HALT	10.2
BITSET	6.6	HIGH	10.2
BOOLEAN	6.1	NEW	10.2
CAP	10.2	NIL	6.7
CARDINAL	6.1	ODD	10.2
CHAR	6.1	PROC	6.8
DEC	10.2	REAL	6.1
DISPOSE	10.2	ROUND	10.2
EXCL	10.2	SIZE	10.2
FALSE	6.1	TRUE	6.1
FLOAT	10.2	TSIZE	10.2
INC	10.2		

5. ОПИСАНИЕ КОНСТАНТ

Описание константы связывает идентификатор с величиной данной константы.

$Описание_константы = идентификатор «=»конст_выражение.
$конст_выражение = простое_конст_выраж [отношение простое_конст_выраж].
$отношение = «=» \ «#» \ «<>» \ «<» \ «>» \ «<=» \ «>=» \ IN.
$простое_конст_выраж = [+ \ —]конст_терм <операция_сложения конст_терм>.
$операция_сложения = «+» \ «—» \ OR.
$конст_терм = конст_фактор<операция_умножения конст_фактор>.
$операция_умножения = «*» \ «/» \ DIV \ MOD \ AND \ «&».
$конст_фактор = квалидент \ число \ строка \ множество \ «(»конст_выражение«)» \ NOTконст_фактор.
$множество = [квалидент]«<»[элемент<«,»элемент>]«>».
$элемент = конст_выражение [«. .»конст_выражение].

Смысл операторов объясняется в р. 9. Идентификатор, предшествующий левой скобке в множестве, указывает тип множества. Если он опущен, то предполагается стандартный тип BITSET (см. п. 6.6).

Примеры описания констант:

```
N=100
LIMIT=2*N-1
ALL=<0..WORDSIZE-1>
S=<0, 1, 3..7, 15>
```

6. ОПИСАНИЕ ТИПОВ

Тип данных определяет набор величин, которые могут принимать переменные этого типа. Служит для привязки идентификаторов к типу. Типом задается также структура переменной. В языке определены три разные структуры: массивы (ARRAY), записи (RECORD) и множества (SET).

```
$Описание—типа  =идентификатор «=» тип.
$тип            =простой—тип \тип—массива \тип— записи
$                \тип—множества \тип указателя
$                \тип процедуры.
$простой—тип  =квалидент \перечисление \тип—диапазон.
```

Примеры:

```
COLOR     =(RED,GREEN,BLUE)
INDEX     =(1..80)
CARD      =ARRAY INDEX OF CHAR
NODE      =RECORD KEY: CARDINAL; LEFT, RIGHT:TREEPTR END
TINT      =SET OF COLOR
TRREPTR   =POINTER TO NODE
FUNCTION  =PROCEDURE (CARDINAL):CARDINAL
```

6.1. Основные типы. Основные типы являются предописанными и обозначаются стандартными идентификаторами:

INTEGER — переменная типа INTEGER рассматривается, как целая величина в диапазоне между MININT и MAXINT.

CARDINAL — переменная типа CARDINAL принимает целые значения между 0 и MAXCARD.

BOOLEAN — переменная этого типа имеет всего два значения TRUE и FALSE. Ее можно записать как перечисления BOOLEAN= (FALSE, TRUE).

CHAR — переменная этого типа является байтовой и принимает значения, соответствующие величинам знаков кода КОИ-7.

REAL — переменные этого типа принимают значения вещественных чисел.

При реализации на 16-битовой ЭВМ, MININT=32768, MAXINT=32767, MAXCARD=65535.

6.2. Тип перечисление (ENUMERATIONS)

Перечисление — это список идентификаторов, которые являются значениями величин данного типа. Эти идентификаторы используются как константы в программе. Только эти константы могут быть значениями переменных типа перечисление. Эти константы упорядочены и отношение упорядоченности задается их последовательностью в перечислении. Для переменных этого типа применимы операции INC и DEC. Значение первой константы в перечислении равно нулю.

```
$Перечисление =«(«список—идентиф»)».
$список—идентиф =идентификатор <«,» идентификатор>.
```

Примеры перечислений:

(CLUB, DIAMOND, HEART, SRADE) (валет, дама, король, туз)

6.3. Тип диапазона

Тип T можно определить как диапазон от основного типа (кроме REAL) или типа перечисление. Диапазон задается указанием наименьшего и наибольшего значения.

```
$Тип—диапазон=«[«конст выражение..конст выражение»]».
```

Первая константа указывает нижнюю границу и не должна превосходить верхнюю. Тип T1 для граничных констант называется базовым типом для T и все операции, применимые к операндам типа T1, также применимы к операциям типа T. Однако, величины, предписанные к переменным типа диапазон, должны лежать внутри заданного интервала. Если нижняя граница неотрицательное число, то базовым типом диапазона будет тип CARDINAL; если же отрицательная величина, то тип — INTEGER.

Говорят, что тип T1 совместим с типом T0, если T1=T0 или T0 является диапазоном базового типа T1 или T0 и T1 диапазоны одного и того же базового типа.

Примеры диапазонов:

```
[0..N-1]   [«A»..«Z»]   [MONDAY..FRIDAY]
```

6.4. Тип массива (ARRAY)

Массив — это структура, состоящая из фиксированного числа компонент, каждая из которых имеет один и тот же

тип. Элементы массива обозначаются индексами, т. е. величинами, соответствующими так называемому индексному типу. Описание массива указывает как тип компоненты, так и тип индекса. Последний может быть перечислением, диапазоном или одним из основных типов BOOLEAN или CHAR.

$Тип_массива=ARRAY простой_тип <«,» простой_тип> OF тип.

Описание:
ARRAY T1, T2, . . , TN OF T
с N индексами T1, T2, . . TN должно рассматриваться, как сокращение описания:
ARRAY T1 OF
ARRAY T2 OF
. . .
ARRAY TN OF T

Примеры массивов:

```
ARRAY [0..N—1] OF CARDINAL
ARRAY [1..10], [1..20] OF [0..99]
ARRAY [—10..10] OF BOOLEAN
ARRAY WEEKDAY OF COLOR
ARRAY COLOR OF WEEKDAY
```

6.5. Тип записи (RECORD)

Запись — это структура, состоящая из фиксированного числа компонент, возможно, различных типов. Описание записи задает каждую компоненту, называемую полем записи. Поле записи состоит из идентификатора и типа идентификатора. Идентификаторы полей записи применяются для описания записи. Они даже используются для ссылок к компонентам записи (см. 8.1).

Существует тип записи с вариантами. Такой тип содержит несколько секций, каждая из которых имеет набор вариантов. Первое поле секции называется тэгом. Первым полем каждого варианта в секции является поле метки. Значение тэга, совпадающее с константой в поле метки для варианта, определяет выбор этого варианта. Метки должны быть константами того типа, который предписан полю тэга.

```
$Тип—записи =RECORD список—полей END.
$список—полей   =поле <«,» поле>.
$поле           =[список—идент «:» тип\
$    CASE [идентификатор «:»]квалидент OF
$    вариант <«  »вариант>[ELSE список—полей] END ].
$вариант        =список—меток «:» список—полей.
$список—меток   =метка <«,» метка>.
$метка          =конст—выражение [«..» конст—выражение].
```


Примеры описаний типа записи:

```
RECORD      DAY:[1..31];
            MONTH:[1..12];
            YEAR:[0..2000]
END
RECORD      имя, отчество:ARRAY [0..9] OF CHAR;
            возраст:[0..99];
            зарплата:REAL
END
RECORD      X, Y:TO;
            CASE TAGO:COLOR OF
            YELLOW..RED:A:TR1;B:TR2
            GREEN:C:TG1;D:TG2
            BLUE:E:TB1;F:TB2
            END;
            Z:TO;
            CASE TAG1:BOOLEAN OF
            TRUE: U, V:INTEGER
            FALSE: R, S:CARDINAL
            END
END
```

Предыдущий пример содержит две секции с вариантами. Вариант первой секции управляется полем TAGO, второй секции полем TAG1.

```
RECORD
            GASE BOOLEAN OF
            TRUE: I:INTEGER
            FALSE: R:CARDINAL
            END
END
```

В этом примере показана запись с вариантами без поля тэга. В этом случае действительное значение варианта, принимаемое переменной, не может быть определено по значению самой переменной. Такая ситуация иногда полезна, но ее следует программировать с особой тщательностью.

6.6. Тип множества (SET)

Этот тип определяется выражением «SET OF T» и в нем представляется сразу все множество величин базового типа T. Тип множества должен быть диапазоном целых чисел из интервала [0..WORDSIZE—1], или типом перечисления с размером не более WORDSIZE—1.

$Тип—множества =SET OF простой—тип.

Стандартный тип BITSET определяется как:

```
BITSET  =SET OF [0..WORDSIZE—1]
```

6.7. Тип указателя (POINTER)

Переменной типа указателя «P» приписывается величина указателя на переменную другого типа T. Говорят, что тип P указывает границу T. Конкретная величина указателя, как правило, генерируется стандартной процедурой NEW или ADR (см. 10.2).

$Тип—указателя=POINTER TO тип.

Кроме указанных величин, таким переменным может быть присвоено значение NIL, которое рассматривается, как пустой указатель.

6.8. Тип процедуры (PROCEDURE)

Переменным типа процедуры T могут приписываться в качестве их величины сама процедура «P». Формальные параметры P (их типы) должны соответствовать списку формальных параметров T. «P» не может быть локальной процедурой другой процедуры и не может быть стандартной процедурой.

```
$Тип—процедуры   =PROCEDURE [список—форм—пар].
$список—форм—пар =«(«[[VAR]формальный—тип
$                   <«,» [VAR]формальный—тип>]»)»[
                    «:»квалидент].
```

Стандартный тип PROC обозначает процедуру без параметров.

```
PROC =PROCEDURE
```

Пример:

```
PR1, PR2:PROCEDURE (CARDINAL, INTEGER)
```

Процедуры PR1 и PR2 могут использоваться в модуле определений вместо такого описания:

```
PROCEDURE PR1(A:CARDINAL, B:INTEGER);
PROCEDURE PR2(A:CARDINAL, B:INTEGER);
```

```
X IN (S1/S2)  =  (X IN S1) # (X IN S2)
```

7. ОПИСАНИЕ ПЕРЕМЕННЫХ

Описание переменных предназначено для того, чтобы ввести переменные и связать их с определенными идентификаторами соответствующего типа и структуры.

$Описание—переменных=список—идентификаторов «:» тип.

Тип определяет набор величин, которые могут принимать переменные, а также операции, которые к ним применяются. Кроме того, тип определяет структуру переменных.

Примеры описаний переменных:

```
I,J:CARDINAL
K:INTEGER
```



```
P,Q:BOOLEAN
S:BITSET
F:FUNCTION
A:ARRAY INDEX OF CARDINAL
W:ARRAY [0..7] OF
   RECORD CH:CHAR;
      COUNT:CARDINAL
   END
T:TREEPTR
```

8. ВЫРАЖЕНИЯ

Выражения — это конструкции, представляющие собой правила вычисления новых значений переменных. Выражения состоят из операндов и операций. Скобки используются для изменения порядка выполнения операций в выражениях.

8.1. Операнды

За исключением чисел, строк и множеств, операнды описываются через, так называемые, обозначения. Обозначение состоит из идентификатора, указывающего на константу, переменную или процедуру. Такой идентификатор может быть квалифицирован именем соответствующего модуля (см. р. 4 и 11) или за ним может быть написан идентификатор поля или индекс, если обозначаемый объект элемент структуры. Для массива A обозначение A[E] указывает тот компонент A, индекс которого является значением выражения E. Обозначение в форме A[E1, E2, ... EN] является сокращением такой записи: A [E1] [E2] ... [EN].

Если R это запись, то обозначение R.F ссылается на поле F в записи R.

Для переменных типа указатель (P:POINTER TO T) обозначение P∧ есть переменная типа T.

```
$Обозначение      =квалидент <«.» идентификатор \
$                   «[«список—выражений»]» \«∧»>.
$список—выражений =выражение <«,» выражение>.
```

Если обозначаемый объект переменная, то ее обозначение ссылается к текущему значению этой переменной. Если объект функция или процедура, то обозначение без параметров ссылается на эту процедуру (например, в описаниях или в списках фактических или формальных параметров процедуры). Если есть список параметров (даже пустой), то обозначение подразумевает активацию процедуры и возвращает значения, результирующие вычисления. Фактические пара-

метры (их типы) должны соответствовать формальным, согласно описанию процедуры (см. р. 10).

Примеры обозначений (см. примеры р. 7):

K	(INTEGER)
A[I]	(CARDINAL)
W[3].CH	(CHAR)
T^.KEY	(CARDINAL)
T^.LEFT^.RIGTH	(TREEPTR)

8.2. Операции

Синтаксис выражения определяется старшинством операций. Имеется четыре класса операций. Операция NOT имеет высший приоритет, затем идут мультипликативные операции, далее аддитивные и, наконец, отношения. Последовательность операций одинакового приоритета выполняется слева направо.

```
$Выражение              = простое_выражение[отношение
                          простое_выражение].
$простое_выражение      = [«+»\«—»]терм <операция_сложения терм>
$терм                   = фактор <операция_умножения фактор>.
$фактор                 = число\строка\множество \обозначение
$                         [фактич_параметры]\«(»выражение«)»\NOT
                          фактор.
$фактич_параметры       = «(»[список_выражений]«)».
```

Список используемых операций приведен ниже. В некоторых случаях различные операции обозначаются одинаковым символом. Тогда действительная операция определяется типом операнда.

8.2.1. Арифметические операции:

+	сложение
—	вычитание
*	умножение
/	вещественное деление
DIV	целое деление
MOD	остаток от целого деления (число по модулю)

Эти операции (кроме /) применимы к операндам типов INTEGER, CARDINAL и ДИАПАЗОН. Если оба операнда имеют тип CARDINAL или ДИАПАЗОН с базовым типом CARDINAL, то результатом будет величина типа CARDINAL. Если оба операнда имеют тип INTEGER или ДИАПАЗОН с базовым типом INTEGER, то и результат будет типа INTEGER.

Операции +, —, * применимы и к операндам типа REAL. В этом случае оба операнда и результат должны быть типа REAL. Операция / применима только к операндам типа REAL.

Для операций сложения и вычитания в типах REAL и INTEGER можно использовать выражения с одним операндом.

Операции DIV и MOD определяются следующими правилами:

X DIV Y — равно целой части от X/Y

X MOD Y — равно остатку от деления X DIV Y

X=(X DIV Y)*Y+(X MOD Y)

остаток от деления имеет знак первого операнда.

—5 DIV —2=2 —5 DIV 2=—2

—5 MOD —2=—1 —5 MOD 2=—1

8.2.2. Логические операции:

OR	логическое ИЛИ
AND	логическое И
NOT	отрицание

Эти операции применимы к операндам типа BOOLEAN и результат тоже типа BOOLEAN.

P OR Q — означает «если P, то TRUE, иначе Q»

P AND Q — означает «если P, то Q, иначе FALSE».

8.2.3. Операции с множествами:

+	объединение множеств
—	вычитание множеств
*	пересечение множеств
/	симметрическая разность

Эти операции применимы к любым операндам типа множества и дают в результате тот же самый тип.

X IN (S1+S2) = (X IN S1) OR (X IN S2)

X IN (S1—S2) = (X IN S1) AND NOT (X IN S2)

X IN (S1*S2) = (X IN S1) AND (X IN S2)

X IN (S1/S2) = (X IN S1)#(X IN S2)

8.2.4. Отношения

Результатом вычисления отношений является тип BOOLEAN. Отношения упорядочивания применимы к базовым типам: INTEGER, CARDINAL, BOOLEAN, CHAR, REAL, ПЕРЕЧИСЛЕНИЕ и тип ДИАПАЗОН.

=	Равно
#	Не равно
<	Меньше
<=	Меньше равно (содержится в множестве)
>	Больше
>=	Больше равно (содержит множество)
IN	содержится в (элемент множества)

Отношения = и # также применимы к множествам и указателям. Применимые к множествам отношения <= и >= означают соответствующее включение. Отношение IN

означает элемент множества. В выражении X IN S, S должно быть типа SET OF T, где T тип совместимый с X.

Примеры выражений (см. примеры р. 7):

1980	CARDINAL
K DIV 3	INTEGER
NOT P OR Q	BOOLEAN
(I+J)*(I—J)	CARDINAL
S— {8, 9, 13}	BITSET
A[I]+A[J]	CARDINAL
A[I+J]*A[I—J]	CARDINAL
(0<=K)&(K<100)	BOOLEAN
T^.KEY=0	BOOLEAN
{13..15}<=S	BOOLEAN
I IN {0,5..8,15}	BOOLEAN

9. ОПЕРАТОРЫ

Операторы — это действия. Операторы бывают элементарные и структурные. Элементарные не состоят из частей, которые сами являются операторами. Таковыми являются: присваивание вызов процедуры, возврат из процедуры и оператор выхода (EXIT). Структурные операторы, наоборот, состоят из частей являющихся операторами. Они используются для выражения последовательного, условного, выборочного или повторяющегося выполнения:

```
$оператор  =[присваивание \ вызов_процедуры \ оператор_IF \
$          оператор_CASE \ оператор_WHILE \ оператор_REPEAT \
$          оператор_LOOP \ оператор_FOR \ оператор_WITH \
$          EXIT \ RETURN [выражение]].
```

Оператор также может быть пустым, что означает отсутствие действия.

9.1. Присваивание

Присваивание предназначено для замены текущей величины обозначения на новую величину, полученную из выражения. Оператор присваивания записывается как «:=» и произносится «становится».

$Присваивание=обозначение «:=» выражение.

Обозначение слева от оператора присваивания является переменной. После присваивания переменная получает значение вычисленного выражения. Старое значение при этом теряется. Тип переменной должен совпадать с типом выражения. Говорят, что операнды совместимы по присваиванию, если их типы совпадают или являются типами INTEGER или CARDINAL или ДИАПАЗОН с такими базовыми типами. Строка длины N1 может быть присвоена строковой переменной длины N2>N1. При этом остаток строки заполняется нулями (OC).

Примеры присваиваний:

I:=K
P:=J=I
J:=LOG2(I+J)
F:=LOG2
S:={2,3,5,7,11,13}
A[I]:=(I+J)*(I—J)
T^.KEY:=I
W[I+1].CH:=«A»

9.2. Вызовы процедур

Вызов процедуры служит для ее активации. Вызов процедуры может содержать список фактических параметров, которые заменяют соответствующие формальные параметры, определенные в описании процедуры (см. р. 10). Соответствие устанавливается позицией параметра в списке фактических и формальных параметров соответственно. Существует два вида параметров: по наименованию и по значению. В случае параметра по наименованию фактический параметр должен быть обозначением переменной. Если он является компонентой массива, то индекс оценивается во время замещения формальных параметров фактическими. Если же параметр является параметром по значению, то соответствующий фактический параметр должен быть выражением. Результирующая величина этого выражения присваивается формальному параметру, который становится локальной переменной. Типы соответствующих формальных и фактических параметров должны совпадать в случае параметров по наименованию и быть совместимы по присвоению в случае параметров по значению.

$Вызов_процедуры=обозначение[фактич_параметры].

Примеры вызовов процедур:

READ(I)
WRITE(J*2+1,6)
INC(A[I])

9.3. Последовательность операторов

Последовательность операторов задает порядок действий определяемых структурными операторами. Каждый оператор

отделяется от следующего знаком «;».
$Посл_операторов=оператор <«;» оператор>.

9.4. Оператор IF

```
$Оператор_IF=  IF выражение THEN посл_операторов
$              <ELSIF выражение THEN посл_операторов>
$              [ELSE посл_операторов] END.
```

Выражения, следующие за IF и ELSIF должны иметь тип BOOLEAN. Эти выражения оцениваются одно за другим, пока какое-нибудь из них не получит значение TRUE. Тогда выполняется связанная с этим условием последовательность операторов. Если в операторе есть ELSE, то связанная с ним последовательность операторов выполняется, если все выражения примут значение FALSE.

Пример:

```
IF (CH>=«A»&(CH<=«Z») THEN READIDENTIFIER
ELSIF (CH>=«0»)&(CH<=«9») THEN READNUMBER
ELSIF CH='»' THEN READSTRING('»')
ELSIF CH=« ' » THEN READSTRING(« ' »)
ELSE SPECIALCHARACTER
END
```

9.5. Оператор CASE

Оператор CASE определяет выбор и выполнение последовательности операторов, связанной с величиной выражения. Происходит это так: вычисляется выражение при CASE и затем выполняется последовательность операторов, отмеченная величиной равной вычисленному выражению.

Тип выражения в CASE должен быть основным (кроме типа REAL), ПЕРЕЧИСЛЕНИЕ или ДИАПАЗОН. Метки должны быть совместимы с этим типом. Не должно быть одинаковых значений в метках CASE. Если вычисленная величина не встречается ни в одной метке, то выполняется последовательность операторов следующая за символом ELSE.

```
SОператор_CASE      =CASE выражение OF CASE <« » CASE>
$                     [ELSE посл_операторов] END.
$CASE               =список_меток «:» посл_операторов.
```

Пример:

```
CASE I OF
   0:P:=P OR Q; X:=X+Y
   1:P:=P OR Q; X:=X-Y
   2:P:=P AND Q; X:=X*Y
END
```

9.6. Оператор WHILE

Оператор WHILE определяет повторяемую последовательность операторов. Выражение типа BOOLEAN оценивается перед каждым выполнением последовательности операторов. Выполнение прекращается, как только оценка выражения даст величину FALSE.

$Оператор_WHILE=WHILE выражение DO посл_операторов END.

Примеры:

```
WHILE J>0 DO
   J:=J DIV 2;I:=I+1;
   WHILE I#J DO
     IF I>J THEN I:=I-J
            ELSE J:=J-I
   END
END;
WHILE(T#NIL)&(T^.KEY#I) DO
T:=T^.LEFT
END
END
```

9.7. Оператор REPEAT определяет повторяемую последовательность операторов. Выражение оценивается после выполнения последовательности. Выполнение прекращается, когда значение выражения станет равно TRUE. Таким образом, последовательность операторов выполняется хотя бы один раз.
$ Оператор_REPEAT=REPEAT посл_операторов UNTIL выражение.

Пример:

```
REPEAT K:=I MOD J; I:=J; J:=K
UNTIL J=0
```

9.8. Оператор FOR

Оператор FOR предназначен для выполнения последовательности операторов, повторяя ее до тех пор, пока значение управляющей переменной не превысит заданного значения. Управляющая переменная изменяется в арифметической прогрессии. Управляющая переменная не может быть элементом массива или записи. Она не может импортироваться и не может быть параметром процедуры. Ее величина не должна изменяться выполняемой последовательностью.

```
$ Оператор_FOR=FOR идентификатор«:=» выражение TO выражение
$ [BY конст_выражение] DO посл_операторов END.
```

Оператор FOR V:=A TO B BY C DO SS END означает повторение вычисления оператора SS с V, последовательно принимающим значения A, A+C, A+2C, ..., A+NC, где A+NC, последнее значение не превышающее B.

Если C отрицательно, то вычисления продолжаются до тех пор, пока A+NC не станет меньше B. V — управляющая переменная, A — начальное значение, B — предел и C — шаг приращения. A и B должны быть совместны по присваиванию с V; C — должно быть константой типа INTEGER или CARDINAL. Если шаг не указан, подразумевается 1.

Примеры:

```
FOR I:=1 TO 80 DO J:=J+A[I] END
FOR I:=80 TO 2 BY —1 DO A[I]:=A[I — 1] END
```

9.9. Оператор цикла LOOP

Оператор LOOP определяет выполнение повторяющейся последовательности операторов. Выполнение заканчивается при прохождении любого оператора EXIT внутри этой последовательности.

$ Оператор_LOOP=LOOP посл_операторов END.

Пример:

```
LOOP
 IF T1^.KEY>X THEN T2:=T1^.LEFT; P:=TRUE
 ELSE T2:=T1^.RIGTH; P:=FALSE END;
      IF T2=NIL THEN EXIT END;
      T1:=T2
  END
```

Операторы WHITH, REPEAT, и FOR могут быть выражены через оператор LOOP, содержащий единственный оператор EXIT. Но их использование удобно, так как они характеризуют наиболее часто встречаемые ситуации, когда окончание повторяемой последовательности зависит от единственного условия, проверяемого в начале или в конце цикла или от достижения предела арифметической прогрессии. Оператор LOOP используется в тех случаях, когда необходимо выразить непрерывное повторение циклического процесса без окончания. Он также полезен в случаях, когда условие выхода проверяется в середине последовательности операторов. Оператор EXIT является контекстуальной, хотя и не синтаксической границей оператора LOOP.

9.10. Оператор WITH

Оператор WITH определяет переменную типа записи и последовательность операторов. В этой последовательности квалифицирование полей записи можно опускать, если квалифицируемая запись-обозначение из оператора WITH. Если

обозначение компоненты массива, то его индекс вычисляется перед последовательностью операторов.

$ Оператор_WITH=WITH обозначение DO посл_операторов END

Пример:

```
WITH T^ DO
    KEY : =0; LEFT:=NIL; RIGTH:=NIL
END
```

9.11. Операторы RETURN и EXIT

Оператор RETURN состоит из ключевого слова и, возможно, следующего за ним выражения E. Эта конструкция означает прекращение работы процедуры (или тела модуля). Выражение E — определяет результат, возвращаемый в случае функции. Его тип должен быть совместим с типом, указанным в заголовке процедуры (см. р. 10). Процедура-функция требует наличия оператора RETURN, указывающего значение результата. Операторов RETURN может быть несколько в одной процедуре, хотя выполняться каждый раз будет только один, в процедурах оператор RETURN определяет конец выполнения тела процедуры. Кроме того, этот оператор является дополнительной, возможно зависящей от условий, точкой окончания программы.

Оператор EXIT состоит из единственного слова EXIT и прекращает выполнение цикла LOOP, продолжение выполнения начинается после END, закрывающего этот оператор LOOP.

10. ОПИСАНИЕ ПРОЦЕДУР

Описание процедуры состоит из заголовка процедуры и блока, который называется телом процедуры. Заголовок определяет имя процедуры и формальные параметры. Блок состоит из описаний и операторов. Имя процедуры повторяется в конце описания процедуры.

Существует два вида процедур — собственно процедура и процедура-функция. Последние активируются вызовом функции, как составляющей выражения. Результат вычисления выглядит как операнд в выражении. Процедура активируется вызовом процедуры. Процедура-функция отличается от просто процедуры указанием ее типа в описании после списка параметров. В теле процедуры-функции должен быть оператор RETURN, который определяет результат работы функции.

Все константы, переменные, типы, модули и процедуры

объявленные внутри блока составляющего тело процедуры являются локальными. Величины локальных переменных, включая и те, что определены внутри локального модуля, являются неопределенными при входе в процедуру. Описания процедур могут быть вложенными. Каждый объект может быть объявлен на своем уровне вложенности. Если он объявлен локальным в процедуре уровня K, то говорят, что объект находится на уровне K+1. Объекты объявленные в модуле, составляющем единицу компиляции, считаются определенными на уровне ноль (см. р. 14).

Помимо формальных параметров и локальных объектов в процедуре доступны объекты из внешнего окружения (за исключением объектов, имеющих одинаковые с локальными объектами имена). Использование идентификатора процедуры для вызова ее самой из ее тела подразумевает рекурсивную активацию процедуры.

```
$Описание_процедуры   =заголовок_процедуры «;» блок
                        идентификатор.
$Заголовок_процедуры  =PROCEDURE идентификатор
                        [формальные_параметры]
$блок      =<описание> BEGIN посл_операторов END.
$описание =CONST <описание_константы «;»>\
$                        TYPE <описание_типа «;»>\
$          VAR <описание_переменных «;»>\
$                        <описание_процедуры «;»>\
$                        <описание_модуля «;»>.
```

10.1. Формальные параметры

Формальные параметры — это идентификаторы, заменяемые фактическими параметрами в вызове процедуры. Присвоение фактических значений формальным параметрам производится во время вызова процедуры. Существует два вида параметров — по значению и по наименованию. Вид задается в списке формальных параметров ключевым словом VAR.

Параметры по значению являются локальными переменными, которым присваивается результат вычисления соответствующего фактического параметра. Фактический параметр в этом случае может быть выражением.

Параметры по наименованию являются внешними объектами и соответствуют фактическим параметрам, которые должны быть переменными. После вызова и активации процедуры формальные параметры становятся этими переменными.

Параметры по наименованию отмечаются символом VAR в описании формальных параметров. У параметров по значению символ VAR отсутствует.

Формальные параметры являются локальными для процедуры, т. е. их область действия ограничивается телом данной процедуры.

```
$Формальные_параметры  = «<»[описание_форм_пар <«;»
$                        описание_форм_пар>]«>»[«:»квалидент].
$описание_форм_пар     = [VAR]список_идентификаторов«:»
                         формальный_тип.
$формальный_тип        = [ARRAY OF] квалидент.
```

Тип каждого формального параметра указывается в списке параметров. В случае параметра по наименованию он должен совпадать с типом фактического параметра. В случае параметра по значению, тип параметра должен быть совместим по присвоению с типом фактического параметра.

Если тип формального параметра задан в виде ARRAY OF T, причем отсутствует указание границ через индексы, то тип T должен совпадать с типом элемента фактического массива, а его индекс должен находиться в интервале [0...N—1], где N — размерность фактического массива. Формальный массив доступен только поэлементно, или он может являться фактическим параметром типа массива, но соответствующий формальный параметр, в свою очередь, должен быть массивом без индекса (ARRAY OF T).

Процедура-функция без параметров имеет пустой список параметров и вызываться она должна через обозначение, также имеющее пустой список параметров.

ОГРАНИЧЕНИЕ: если формальный параметр имеет тип процедуры, то соответствующий фактический параметр должен быть либо процедурой, объявленной на уровне ноль, либо переменной этого же типа.

Параметр не может быть именем стандартной процедуры (см. п. 10.2).

Примеры описания процедур:

```
READ(X),WRITE(X,N),LOG2(X):CARDINAL
      PROCEDURE READ(VAR X:CARDINAL);
        VAR I:CARDINAL;CH:CHAR;
      BEGIN I:=0;
      REPEAT READCHAR(CH)
      UNTIL (CH>=«0»)&(CH<=«9»);
      REPEAT I:=10*I+(CARDINAL(CH)—CARDINAL(«0»));
          READCHAR(CH)
      UNTIL (CH<«0») OR (CH>«9»);
      X:=I
END READ
PROCEDURE WRITE(X,N:CARDINAL);
```

```
  VAR I:CARDINAL;
      BUF:ARRAY [1..10] OF CARDINAL;
BEGIN I:=0;
  REPEAT INC(I);BUF[I]:=X MOD 10; X:=X DIV 10
  WHILE N>I DO
     WRITECHAR(« »); DEC(N)
  END;
  REPEAT WRITECHAR(CHAR(BUF[I]+CARDINAL(«0»)));
    DEC(I)
  UNTIL I=0;
 END WRITE
PROCEDURE LOG2(X:CARDINAL):CARDINAL;
   VAR Y:CARDINAL; (* X>0 *)
 BEGIN X:=X—1; Y:=0
   WHILE X>0 DO
      X:= X DIV 2; Y:=Y+1
   END;
   RETURN Y
 END LOG2
```

10.2. Стандартные процедуры

Стандартные процедуры считаются предопределенными. Некоторые из этих процедур не могут быть отчетливо описаны, т. к. они применяются к разным типам операндов или имеют несколько возможных списков параметров.

Стандартные процедуры:

ABC(X)	Абсолютная величина. Тип результата = тип аргумента.
ADR(V)	Адрес переменной V (тип CARDINAL).
ASH(X, H)	Арифметический сдвиг. X:=X*(2**N)
CAP(CH)	Если буква строчная, то сделать ее прописной. Только для латинского алфавита.
FLOAT(I)	Перевод целого числа в вещественное (REAL).
HIGH(A)	верхняя граница массива A
ODD(X)	X MOD 2 # 0 (тип BOOLEAN).
ROUND(X)	X округляется до ближайшего целого типа INTEGER.
TSIZE(T)	Размер переменной типа T.
SIZE(X)	Размер переменной X.
DEC(X)	X:=X—1
DEC(X, N)	X:=X—N
EXCL(S, I)	S:=S—<I>
INC(X)	X:=X+1
INC(X, N)	X:=X+N
INCL(S, I)	S:=S+<I>
HALT	Прекратить выполнение программы.

Процедуры типа INC и DEC применимы также к операндам типа ПЕРЕЧИСЛЕНИЕ и CHAR. В этих случаях X заменяется своим N-тым предшественником или последователем.

NEW и DISPOSE содержат обращения к процедурам ALLOCATE и DEALLOCATE, которые либо программируются, либо импортируются из другого модуля.

```
NEW(P)                = ALLOCATE (P,TSIZE(T))
DISPOSE(P)            = DEALLOCATE (P,TSIZE(T))
NEW(P,T1,T2,...)      = ALLOCATE (P,TSIZE(T,T1,T2,...))
DISPOSE(P,T1,T2,...)  = DEALLOCATE(P,TSIZE(T,T1,T2,...))
```

В этих примерах переменная «P» объявлена как: VAR P: POINTER TO T. Сами процедуры должны иметь тип: PROCEDURE(VAR ADDRESS, CARDINAL).

11. МОДУЛИ

Модуль состоит из описаний и последовательности операторов, составляющих тело модуля (блок). Заголовок модуля содержит идентификатор модуля и списки импортных и экспортных объектов.

Список импортных объектов содержит идентификаторы объектов, описанных вне модуля, но используемых внутри модуля. Список экспортных объектов указывает идентификаторы тех объектов, которые определены внутри модуля, но использоваться будут вне его. Таким образом, модуль представляет собой оболочку, окружающую его локальные объекты и их доступность находится под управлением программиста.

Локальные объекты модуля находятся на том же уровне, что и модуль. Их можно рассматривать в качестве локальных объектов для процедуры, содержащей модуль, но помещенных в более ограниченную область.

```
$Описание_модуля = MODULE имя_модуля[приоритет]«;»[импорт]
$                  [экспорт] блок имя_модуля.
$имя_модуля      = идентификатор.
$приоритет       = «[«целое»]».
$экспорт         = EXPORT [QUALIFIED] список_идентификаторов «;».
$импорт          = [FROM имя_модуля] IMPORT список_идентификаторов«;».
```

Последовательность операторов, составляющих тело модуля (блок) выполняется, когда вызывается процедура, для которой этот модуль локальный. Если описано несколько модулей, то их тела выполняются в том порядке, в каком расположены модули. Тела модулей предназначены для инициализации локальных переменных и должны рассматриваться, как предисловия к операторам процедур, входящих в модуль. Ес-

ли идентификатор встречается в списке импортных (экспортных) объектов, то эти объекты могут использоваться вне (внутри) модуля. Однако, если за символом EXPORT следует QUALIFIED, то идентификаторы должны квалифицироваться именем модуля, который их экспортирует. Такое описание называется квалифицированным экспортом и применяется в тех случаях, когда модуль предназначен для использования другими модулями заранее неизвестными. Квалифицированный экспорт служит для предотвращения случаев совпадения имен идентификаторов из различных модулей.

Если список импортных переменных помечен символом FROM и именем модуля, то импортные переменные могут употребляться без квалификации именем модуля, т. е. так, как будто они экспортированы без указания QUALIFIED.

Если экспортируется тип запись, то все ее поля экспортируются тоже без явного объявления в экспортном и импортном списках. Аналогично для идентификаторов констант в списках типа ПЕРЕЧИСЛЕНИЕ.

Стандартные идентификаторы импортируются автоматически. Отметим, что стандартные идентификаторы могут быть переописаны только в процедурах, но не в модулях, даже, если это единица компиляции (см. р. 14).

Примеры описания модулей:

в этом модуле просматривается входной текст и копируется в выводную последовательность знаков. Ввод производится по одному знаку через процедуру INCHR, а познаковый вывод через процедуру OUTCHR. Знаки поступают в коде КОИ-7. Управляющие знаки игнорируются за исключением знака LF и FS (FILE SELECTOR). Они переводятся в пробел и устанавливают булевские переменные EOLN и EOF соответственно. Предполагается, что перед FS записан LF.

```
MODULE LINEINPUT;
   IMPORT INCHR,OUTCHR;
   EXPORT READ,NEWLINE,NEWFILE,EOLN,EOF,LNO;
CONST LF=12C; CR=15C; FS=34C;
VAR LNO:CARDINAL; (* номер строки *)
      CH:CHAR; (* последний считанный знак *)
    EOF,EOLN:BOOLEAN;
    PROCEDURE NEWFILE;
    BEGIN
      IF NOT EOF THEN
        REPEAT INCHR(CH) UNTIL CH=FS;
      END;
    EOF:=FALSE;EOLN:=FALSE; LNO:=0
    END NEWFILE;

PROCEDURE NEWLINE;
```



```
BEGIN
   IF NOT EOLN THEN
     REPEAT INCHR(CH) UNTIL CH=LF;
     OUTCHR(CH);OUTCHR(LF)
   END;
   EOLN:=TRUE; INC(LNO)
PROCEDURE READ (VAR X:CHAR);
BEGIN (* предполагается NOT EOLN AND NOT EOF *)
 LOOP INCHR(CH); OUTCHR(CH);
   IF CH>=«  » THEN
      X:=CH; EXIT
     ELSIF CH=LF THEN
       X:=«  »; EOLN:=TRUE; EXIT
     ELSIF CH=FS THEN
       X:=«  »; EOF:=TRUE; EOLN:=TRUE; EXIT
   END
   END
END READ;
BEGIN EOF:=TRUE; EOLN:=TRUE
END LINEINPUT
```

В следующем примере описан модуль, который управляет резервированием дорожек на диске и защищает их от несанкционированного доступа. Процедура-функция NEWTRACK выдает номер свободной дорожки, которая становится зарезервированной. Дорожки освобождаются вызовом процедуры RETURNTRACK.

```
MODULE TRACKRESERVATION;
 EXPORT NEWTRACK,RETURNTRACK;
  CONST NTR=1024; (* количество дорожек *)
     W=16; (* длина слова *)
     M=NTR DIV W;
  VAR I:CARDINAL;
     FREE:ARRAY [0..M—1] OF BITSET;
  PROCEDURE NEWTRACK( ):INTEGER:
  (* резервирует свободную дорожку и выдает ее
   номер в качестве результата. Если дорожка не
   найдена, результат=—1 *)
  VAR I,J:CARDINAL; FOUND:BOOLEAN;
  BEGIN FOUND:=FALSE; I:=M;
   REPEAT DEC(I); J:=W;
     REPEAT DEC(J);
        IF J IN FREE[I] THEN FOUND:=TRUE END
      UNTIL FOUND OR (J=0)
   UNTIL FOUND OR (I=0);
   IF FOUND THEN EXCL (FREE[I],J); RETURN I*W+J
        ELSE RETURN —1
   END (* IF *)
  END NEWTRACK;

  PROCEDURE RETURNTRACK (K:CARDINAL);
  BEGIN (* предполагаем, что 0<=K<NTR *)
    INCL (FREE[K DIV W], K MOD W)
  END RETURNTRACK;
```

```
BEGIN (* пометим все дорожки свободными *)
    FOR I:0 TO M—1 DO FREE[I]:={0..W—1} END
END TRACKRESERVATION.
```

6 модулях см. также р. 14

12. СИСТЕМНО-ЗАВИСИМЫЕ СВОЙСТВА ЯЗЫКА

MODULA-2 содержит некоторые возможности для программирования нижнего уровня, относящегося к объектам, связанным с данным компьютером и с реализацией. К их числу относятся, например, возможность доступа к устройствам управляемым от ЭВМ, средства для изменения типов данных и т. д. Пользоваться этими средствами следует очень осторожно и полезно ограничить их использование небольшим числом модулей. Многие подобные объекты, описанные как процедуры или типы данных, импортируются из модуля SYSTEM.

ЗАМЕЧАНИЕ: так как объекты модуля SYSTEM подчиняются специальным правилам, они должны быть известны компилятору. Поэтому модуль SYSTEM называется псевдомодулем и не имеет модуля определений (см. р. 14).

Из модуля SYSTEM экспортируются типы: WORD, ADDRESS, PROCESS, процедуры: NEWPROCESS, TRANSFER, IOTRANSFER и, возможно, **другие объекты, зависящие** от используемой операционной системы (см. р. 13).

Тип WORD представляет отдельно доступную **единицу** памяти. Никакие операции, кроме присвоения не определены для этого типа. Однако, если формальный параметр процедуры имеет тип WORD, то соответствующий фактический параметр может иметь любой тип, использующий одно слово для хранения в памяти. Следовательно, это могут быть типы CARDINAL, INTEGER, BITSET и любые указатели. Если **формальный параметр имеет тип** ARRAY OF WORD, это соответствует фактическому параметру любого типа. Например, это может быть запись, интерпретируемая, как массив слов.

Тип ADDRESS определяется так:

```
ADDRESS=POINTER TO WORD
```

Этот тип совместим со всеми типами указателей и с типом CARDINAL. В силу этого все операторы целой арифметики применимы к данному типу. Таким образом, тип ADDRESS может использоваться для вычисления адресов и выдачи указателей в качестве результата.

В следующем примере показан простой диспетчер памяти, демонстрирующий типичное использование типа ADDRESS.

```
MODULE STORAGE;
    FROM SYSTEM IMPORT ADDRESS;
    EXPORT ALLOCATE;
    VAR LASTUSED:ADDRESS;
    PROCEDURE ALLOCATE (VAR A:ADDRESS; N:CARDINAL);
    BEGIN A:=LASTUSED; INC(LASTUSED,N)
    END ALLOCATE;
BEGIN LASTUSED:=0
END STORAGE.
```

Помимо этих объектов, экспортируемых из SYSTEM, существуют два других системно-зависимых свойства.

Первое свойство дает возможность использования типа идентификатора «T», как имени функции пересылки типа из типа операнда в тип T. Ясно, что такая функция зависит от представления данных и содержит инструкции для преобразований.

Второе свойство касается описаний переменных и позволяет определять абсолютные адреса переменных, а также изменять распределение данных, сделанное компилятором. Это свойство используется в первую очередь для доступа к специальным ячейкам, таким, как регистры устройств. Абсолютные адреса определяются как константы, заключенные в квадратные скобки и следующие за идентификатором переменной в описании. Выбор соответствующего типа данных предоставляется программисту.

Примеры (см. также п. 13.2):

```
VAR TWS[177564B]:BITSET;(* статус консоли *)
    TWB[177566B]:CHAR;  (* буфер консоли *)
```

13. ПРОЦЕССЫ

MODULA-2 спроектирована для реализации на однопроцессорном компьютере. Для мультипрограммирования имеются лишь несколько основных возможностей, которые позволяют определять квазипараллельные процессы и действительную параллельность для внешних устройств. Сопрограммы это процессы, которые выполняются на единственном процессоре.

13.1. Создание процессов и передача управления.

Новый процесс создается вызовом процедуры:

```
PROCEDURE NEWPROCESS(P:PROC;A:ADDRESS;N:CARDINAL;P1:PROCESS)
```

P — процедура, задающая процесс.
A — адрес рабочей области.
N — размер этой рабочей области.

Новый процесс с «P» в качестве программы и рабочей областью A, размером N(слов) назначается для P1. Этот процесс распределяется, но не активируется. P должна быть процедурой, объявленной на уровне 0. Передача управления между двумя процессами осуществляется через вызов процедуры:

PROCEDURE TRANSFER (VAR P1,P2:PROCESS)

При этом приостанавливается выполнение текущего процесса. Информация о процессе запоминается в P1, а из P2 восстанавливается информация о процессе, записанном туда процедурами TRANSFER или IOTRANSFER. Присвоение P1 производится после идентификации нового процесса P2: следовательно, фактические параметры могут быть одинаковы. Очевидно, что P2 должен уже хранить информацию о процессе, что делается либо вызовом NEWPROCESS, либо TRANSFER. Обе процедуры, как и тип PROCESS должны быть импортированы из модуля SYSTEM. Программа заканчивает выполнение, когда достигается конец процедуры, которая есть тело процесса. Так как операции для процессов не определены, а присвоения занимают много времени, то с ними лучше всего работать через указатели.

13.2. Процессы для устройств и прерывания.

Если процесс управляет внешним устройством, то процессор может быть передан другому процессу после того, как операция на устройстве запущена, что приводит к параллельному выполнению этого другого процесса и процесса устройства. Обычно прекращение операции на устройстве вызывает прерывание процессора. В понятиях языка MODULA-2 прерывание есть операция TRANSFER. Эта передача управления при прерывании заранее программируется и комбинируется с передачей после инициализации устройства. Эта комбинация выражается обращением к процедуре:

```
PROCEDURE IOTRANSFER (VAR P1,P2:PROCESS; VECTOR:CARDINAL)
```

По аналогии с TRANSFER этот вызов приостанавливает процесс обратившийся к устройству, назначает его для P1, возобновляет процесс P2 и, кроме того, по прерыванию, свидетельствующему об окончании работы устройства, назначает прерванный процесс снова на P2 и возобновляет процесс P1.

VECTOR — это адрес вектора устройства. Процедура IOTRANSFER также должна быть импортирована из модуля SYSTEM.

Необходимо, чтобы прерывания можно было запрещать на определенное время, например, когда идет работа с общими переменными взаимодействующих процессов или, когда выполняются критические по времени участки программы. Поэтому каждому модулю приписывается определенный уровень приоритета и каждому устройству, которое может вызвать прерывания, тоже назначается (аппаратно) уровень приоритета. Выполнение программы модуля может быть прервано тогда и только тогда, когда приоритет устройства выше приоритета модуля. Поскольку приоритет модуля устанавливается программно в интервале [0..7] то, задавая его выше или ниже приоритета устройства, можно блокировать или разрешать прерывания от этого устройства. Если явное описание приоритета опущено, приоритет процедуры равен приоритету вызвавшей задачи.

```
для N знаков.
 MODULE TYPEWRITER[4];  (*четвертый приоритет*)
 FROM SYSTEM IMPORT WORD,PROCESS,NEWPROCESS,TRANSFER,
 IOTRANSFER,LISTEN;
  EXPORT TYPEOUT;
  CONST M=32;
 VAR N:INTEGER;  (*количество знаков в буфере*)
    B:ARRAY [1..M] OF CHAR;
    PRO:PROCESS; (*PRODUSER *)
  CON:PROCESS;  (* CONSUMER = драйвер телетайпа*)
  WSP:ARRAY [0..20] OF WORD;
  TWS[177564B]:BITSET; (*регистр состояния*)
 TWB[177566B]:CHAR;  (*буферный регистр*)
 PROCEDURE TYPEOUT (CH:CHAR);
  BEGIN INC(N);
    WHILE N>M DO LISTEN END;
    B[IN]:=CH; IN:=IN MOD M +1;
   IF N=0 THEN TRANSFER(PRO,CON) END
 END TYPEOUT;
 PROCEDURE DRIVER;
 BEGIN
   LOOP
      DEC(N);
      IF N<0 THEN TRANSFER(CON,PRO) END;
     TWB:=B[OUT]; OUT:=OUT MOD N + 1;
      TWS:=<6>; IOTRANSFER(CON,PRO,64B); TWS:=<>
   END
  END DRIVER;
BEGIN N:=0; IN:=1; OUT:=1;
 NEWPROCESS (DRIVER,ADR(WSP),SIZE(WSP),CON);
 TRANSFER(PRO,CON)
END TYPEWRITER.
```

В вышеуказанном примере показан модуль с процессами, работающими как драйвер для телетайпа. В модуле определен буфер B для N знаков.

LISTEN это процедура, которая снижает приоритет процессора до уровня, который позволяет возникшему прерыванию быть обработанным процессором.

14. ЕДИНИЦА КОМПИЛЯЦИИ

Текст, обрабатываемый компилятором как целое, называется единицей компиляции. Существует три вида единиц компиляции: главный модуль, модуль определений и модуль реализаций. Главный модуль состоит из главной программы и из программных модулей. Его отличием от других типов модулей является отсутствие экспортного списка. Импортируемые объекты определяются в других (отдельно компилируемых) частях программы.

Модуль определений задает имена и свойства объектов, которые важны для внешнего пользователя, т. е. других модулей, импортирующих эти объекты.

Модуль реализации содержит локальные объекты и операторы, знание которых не нужно потребителю.

Модуль определений содержит список экспортируемых объектов, т. е. констант, типов, переменных и процедур, а также их описания. Кроме того, для процедур приводятся заголовки процедур. Соответствующий модуль реализации содержит полное описание процедур и, возможно, другие описания не экспортируемых объектов. Модули определений и реализаций существуют в паре. Оба могут содержать список объектов и все объекты, объявленные в модуле определений, доступны в модуле реализаций без объявлений импорта.

```
$Модуль_определений = DEFINITION MODULE имя_модуля ";"
$                     <импорт><экспорт> <описание> END
$                     имя_модуля.
$описание           = CONST<описание_константы ";">\
$                     TYPE< идентификатор["="тип] ";">\
$                     VAR < описание_переменных>\
$                     <заголовок_процедуры";">.
$программный_модуль = MODULE имя_модуля[приоритет]";"
$                     <импорт><блок> имя_модуля ".".
$единица_компиляции = модуль_определений\ [IMPLEMENTATION]
$                     программный_модуль ".".
```

Очевидно, что модуль определений представляет собой интерфейс модульной пары определения/реализации. Модуль

определений требует квалифицированного экспорта. Определение типа может состоять из полной спецификации типа или только из идентификатора типа. В первом случае экспорт называется прямым, во втором ограниченным. При ограниченном экспорте полная спецификация должна появиться в соответствующем модуле реализации. Если импортируемый тип известен лишь по имени, то все его свойства скрыты. Поэтому, процедуры, работающие с операндами этого типа и, в особенности, работающие с компонентами типа, должны быть определены в том же самом модуле, который скрывает свойства типа. Ограниченный экспорт, как правило, применяется для указателей и диапазонов стандартных типов. Примеры единиц компиляции приведены в р. 16.

15. РЕАЛИЗАЦИЯ И ИСПОЛЬЗОВАНИЕ ЯЗЫКА MODULA-2

В этом разделе описана реализация языка MODULA-2. Эта система состоит из компилятора, компоновщика, отладчика и базового монитора, включающего загрузчик. Единицей данных, которая обрабатывается этой системой, является файл в том виде, как он определен в соответствующей системе.

Базовый монитор — это резидентная программа, называемая MODULA. Она распознает имя файла (расширение по умолчанию LOD) и, используя свой загрузчик, загружает указанный файл. Этот файл должен быть произведен компоновщиком. После загрузки управление передается программе и по окончанию возвращается в базовый монитор. Готовность монитора принять имя следующей программы указывается выводом «*».

Сам компилятор является программой, написанной на языке MODULA-2 и записан в библиотеку под именем COMP. После загрузки он запрашивает имя файла с единицей компиляции и начинает его обрабатывать. Расширение единицы компиляции по умолчанию MOD. (Рекомендуется расширение MOD для программных модулей и DEF для модулей определений.) Компилятор производит листинг с тем же именем, что и компилируемая программа, и расширением LST. Вывод идет на диск. В случае успешной компиляции программного модуля выводится объектная программа для компоновщика с расширением LNK. После компиляции модулей определений выводится таблица символов с расширением SYM. Также генерируется файл, который необходим при ра-

боте отладчика. Он имеет расширение REF. Кроме входного текста, компилятор требует на входе файлы с таблицей символов (SYM) для всех модулей, указанных в списках импорта компилируемой программы (кроме модуля SYSTEM).

После компиляции объектный код должен быть скомпонован с объектным кодом импортируемых модулей с помощью компоновщика, который также является программой, написанной на языке MODULA-2. (имя LINK). После загрузки компоновщик запрашивает имя программы, называемой главным (MASTER) файлом. В дальнейшем он требует все файлы в объектном коде, которые следует скомпоновать. Расширение имен этих файлов по умолчанию LNK. В результате работы формируется файл с именем главной программы и расширением LOD в загрузочном формате.

В процессе компоновки создается код — образ памяти. Стратегия распределения памяти основана на принципе стека. По умолчанию компануется загрузочный модуль, в который входит базовый монитор. Это позволяет строить оверлейные программы, поскольку загрузчик является частью базового монитора и может вызываться непосредственно из программы. Кроме того в базовый монитор входят некоторые программы ввода-вывода и программы для связи с операционной системой.

Единица компиляции программы — это отдельно компилируемая, но не независимая часть всей программы. Зависимость устанавливается файлом таблицы символов, который необходим компилятору для проверок совместимости типов. Такой принцип работы требует, чтобы модули определений компилировались раньше модулей реализации. Важно, что при этом сам модуль реализации, содержащий импортируемые объекты, может перекомпилироваться. Это не потребует перекомпиляции всех других модулей. При изменении же модуля определений, требуется перекомпиляция модуля реализации и всех модулей, импортирующих объекты из этого модуля.

Для удобства проверки совместимости типов среди компонуемых версий, компилятор записывает в символьный файл (SYM) временную метку (день и время компиляции). Позже, при компиляции модуля реализации, эта метка копируется в объектный файл (LNK). Если нарушена хронологическая последовательность создания модулей (сначала модуль определений, затем модуль реализации) компилятор распознает это и отвергает такие файлы.

Если программа заканчивается с ошибкой, то ее состояние

может быть изучено с помощью отладчика, который также написан на языке MODULA-2 и называется DEBUG. Будучи вызвана, она требует ввода имени программы и наличия двух соответствующих файлов с расширением LST и REF. Состояние самой программы сохраняется в файле (дампе памяти), создаваемом при ошибочном завершении программы.

Большинство программ на языке MODULA-2 использует предварительно скомпилированные модули, импортируя их. Поэтому реализация это не только базовый монитор, компилятор и т. д., но и набор часто используемых утилит. В их число входят процедуры создания, чтения, записи и удаления файлов, обычные преобразования ввода-вывода для работы с внешними устройствами и другие процедуры. Несколько таких модулей описано в следующей главе. Тем не менее, хотя эти модули и существенны для большинства программ, они не являются частью языковых определений. В принципе, любой программист волен написать собственные версии таких модулей, создавая, тем самым, свой набор утилит или даже включая такие модули в определение языка. Подобная гибкость языка есть результат модульной концепции.

16. МОДУЛИ СТАНДАРТНЫХ УТИЛИТ

В этом разделе описан набор модулей, которые часто используются в программах. Для стандартизации программ на языке MODULA-2 желательно иметь подобные унифицированные модули в любой реализации. В набор входят процедуры ввода-вывода, поддержка квазипараллельных процессов и др.

16.1. Ввод-вывод

Процедуры, содержащиеся в модуле InOut, используются для выполнения ввода и вывода на стандартных устройствах или последовательных файлах. В частности, эти процедуры выполняют необходимые преобразования данных между стандартными типами CARDINAL и INTEGER и последовательностью знаков. Процедуры ввода читают со стандартного вводного устройства системы (консоли). Туда же идет вывод из процедур записи. Для вывода на другие устройства следует воспользоваться процедурой OpenIO. Тогда эта процедура запрашивает спецификации имен файлов. Ввод-вывод вновь назначаются на консоль после выполнения процедуры CloseIO, которая также производит закрытие файлов. В некоторых процедурах указывается количество знаков записи — «N». Если N>M — число знаков необходимых для представ-

ления величины X, то N минус M пробелов предшествует X. Если N<M, то указание N игнорируется.

```
DEFINITION MODULE InOut;
FROM SYSTEM IMPORT WORD;
EXPORT QUALIFIED
   StrLeng, String, OpenIO, CloseIO, Read, ReadInt,
  Write, WriteInt, WriteCard, WriteOct, WriteHex,
   WriteLn, WriteString, ShowString;
CONST StrLeng = 80;
TYPE String = ARRAY [0..StrLeng—1] OF CHAR;

PROCEDURE OpenIO;
PROCEDURE CloseIO;
PROCEDURE Read(VAR ch: CHAR);
PROCEDURE ReadInt(VAR x: INTEGER;
                  VAR NextChar: CHAR;
                  VAR IsNum: BOOLEAN);
(* При выполнении процедур InOut следует учитывать, что:
— пробелы и специальные знаки пропускаются (игнорируются);
— если вводится последовательность цифр (возможно со знаком), то она читается, как десятичное целое;
— переменная IsNum=TRUE, если прочитано число;
— NextChar содержит последний введенный символ, т. е. тот, который записан за последовательностью цифр;
— OC указывает конец вводной последовательности;
— проверок не производится *)
   PROCEDURE WriteInt(x: INTEGER; n: CARDINAL);
   PROCEDURE WriteCard(x, n: CARDINAL);
   PROCEDURE WriteOct(w: WORD; n: CARDINAL);
   PROCEDURE WriteHex(w: WORD; n: CARDINAL);
   PROCEDURE WriteLn;
   PROCEDURE WriteString(s: String);
   PROCEDURE ShowString(s: String; cr: BOOLEAN);
END InOut.
```

16.2. Потоки

Этот модуль содержит процедуры для работы с последовательными файлами. В нем определены последовательности знаков и слов, называемые потоками. Для того, чтобы начать работать с потоком, его следует связать с файлом, который в свою очередь должен быть открыт. Когда поток связан с файлом, он может быть прочитан или записан. Тип элемента потока определяется во время связывания потока с файлом. Потоки читаются через процедуры ReadChar и WriteChar (для слов ReadWord и WriteWord). Каждый вызов читает или пишет очередной элемент.

Запись потока завершается вызовом EndWrite. После этого поток может быть отсоединен от файла и файл закрыт. При чтении элемента булевская переменная, полученная из процедуры EOS (END OF STREAM) указывает: может ли следующий элемент быть прочитан или достигнут конец потока. В последнем случае значение из ReadChar будет равно OC. Процедура Reset используется для того, чтобы установить позицию чтения или записи на начало потока.

```
DEFINITION MODULE Streams;
FROM SYSTEM IMPORT WORD;
FROM Files IMPORT FILE;
EXPORT QUALIFIED
   STREAM, Connect, Disconnect, Reset
   WriteWord, WriteChar, EndWrite,
   ReadWord, ReadChar, EOS,
   GetPos, SetPos;
TYPE STREAM;

PROCEDURE Connect(VAR s:STREAM; f:FILE; ws:BOOLEAN);
(*  соединение потока с открытым файлом f.
     f = номер канала в ФОДОС-2, 0<F<16.
     ws = "s поток слов (не знаков)"  *)
PROCEDURE Disconnect(VAR s:STREAM;closefile:BOOLEAN);
PROCEDURE Reset(s: STREAM);
PROCEDURE WriteWord(s: STREAM; w: WORD);
PROCEDURE WriteChar(s: STREAM; ch: CHAR);
PROCEDURE EndWrite(s: STREAM);
PROCEDURE ReadWord(s: STREAM; VAR w: WORD);
PROCEDURE ReadChar(s: STREAM; VAR ch: CHAR);
PROCEDURE EOS(s: STREAM): BOOLEAN;
PROCEDURE GetPos(s: STREAM; VAR highpos,
                 lowpos: CARDINAL);
PROCEDURE SetPos(s:STREAM;highpos,lowpos:CARDINAL);
END Streams.
```

В следующем примере показано типичное использование потоков. Создается поток "OUT", связанный с файлом 1 и поток "IN", связанный с файлом 2.

```
VAR IN,OUT:STREAM; ch:CHAR;
 CONNECT(OUT,1,FALSE);
 REPEAT ... WriteChar(OUT,CH) ... UNTIL ...;
 EndWrite(OUT); DISCONNECT(OUT,FALSE);
 CONNECT(IN,2,FALSE); ReadChar(IN,ch);
 WHILE NOT EOS(IN) DO
 ... READCHAR(IN,CH) ...
 END;
 DISCONNECT(IN,TRUE)
```

16.3. Файлы

Этот модуль позволяет получить доступ к файлам ФОДОС-2 процедуры доступа зависят от лежащей в основе файловой системы, которая является частью операционной системы. Файл в ФОДОС-2 определяется как последовательный и блочный, соответствующий дисковым секторам. Каждый блок состоит из 512 знаков или 256 слов. Блоки нумеруются 0,1,2,...

Каждый файл однозначно определяется переменной типа FILE, которая в ФОДОС-2 называется номером канала. Файлу присваивается канальный номер при обращении к Lookup (если такой файл уже есть в системе) или обращением к Create (когда создается новый файл). Обе процедуры требуют указания имени файла. Существующие файлы можно читать или перезаписывать. Новый файл вводится в оглавление при вызове процедуры Close. Если файл не надо сохранять, то вызывается процедура Release.

В ФОДОС-2 имя файла состоит из 12 символов. Первые три указывают устройство (по умолчанию «DK»), следующие 6 — имя файла и последние три — расширение имени. Для более детального знакомства читатель отсылается к соответствующей литературе по ФОДОС-2.

```
DEFINITION MODULE Files;

  FROM SYSTEM IMPORT ADDRESS, WORD;

  IMPORT SystemTypes;
```



```
EXPORT QUALIFIED FILE, FileName, Lookup, Create,
                 Delete, Release, Close,
                 WriteBlock, ReadBlock, Rename,
                 SetBlock, TransmitBlock, Rad50Name,
                 Radix50, Errcode;
(*  Имя          Соответствующий     Функция
   процедуры     запрос системы
   Lookup          .LOOKUP           поиск файла в оглавлении
   Create          .ENTER            создание нового файла
   Delete          .DELETE           уничтожить имя файла в
                                     оглавлении
   Release         .PURGE            закрыть файл, не занося
                                     имя в оглавление
   Close           .CLOSE            закрыть файл, занести
                                     имя в оглавление
   WriteBlock      .WRITEW           запись
   ReadBlock       .READW            чтение
   Rename          .RENAME           переименовать файл
*)
TYPE FILE = [0..15];
     FileName = SystemTypes.FileName;
                (* ARRAY [0..11] OF CHAR *)

PROCEDURE Lookup(f: FILE; fn: FileName;
                 VAR reply: INTEGER);
  (* найти файл f в оглавлении
     reply: >=0 = найден, длина файла в блоках.
            <0 = ошибка
            -1 = канал занят
            -2 = файл не найден *)

PROCEDURE Create(f: FILE; fn: FileName;
                 VAR reply: INTEGER);
  (* создать новый файл f
     reply: >=0 = создан, длина файла
            <0 = ошибка
            -1 = канал занят
            -2 = нет памяти  *)

PROCEDURE Delete(f: FILE; fn: FileName;
                 VAR reply: INTEGER);
  (* уничтожить файл F и вычеркнуть имя из оглавления
     reply: >=0 = выполнено, длина файла
            <0 = ошибка
            -1 = канал занят
            -2 = файл не найден  *)

PROCEDURE Close(f: FILE);
  (* закрыть файл F и записать его имя в оглавление*)
PROCEDURE Release(f: FILE);
  (* закрыть файл не записывая имя в оглавление  *)

PROCEDURE ReadBlock(f: FILE; p: ADDRESS; blknr,
                    wcount: CARDINAL;
                    VAR reply: INTEGER);
```

```
  (* чтение файла f
     p:       адрес буфера
     blknr:   номер блока, с которого начинать чтение
     wcount:  количество слов, которые следует прочитать
     reply:   >=0 = количество прочитанных слов
               <0 = ошибка
               -1 = сбой оборудования
               -2 = канал не открыт  *)

PROCEDURE WriteBlock(f: FILE; p: ADDRESS; blknr,
                     wcount: CARDINAL;
                     VAR reply: INTEGER);
  (* запись в файл f
     p:       адрес буфера
     blknr:   номер первого блока, с которого начинать
              запись
     wcount:  количество слов, которые следует записать
     reply:   >=0 = количество записанных слов
               <0 = ошибка
               -1 = сбой оборудования
               -2 = канал не открыт  *)
PROCEDURE Rename(f: FILE; new, old: FileName;
                 VAR reply: INTEGER);
  (* переименовать файл f , (не должен быть открытым)
     reply:   0 = выполнено
             <0 = ошибка
             -1 = канал занят
             -2 = файл не найден  *)
(*-------------------------------------------------------*)
TYPE Rad50Name = ARRAY [0..3] OF INTEGER;
VAR Errcode[52B]: CHAR; (* в ФОДОС-2 здесь хранится
                           код произошедшей ошибки *)
PROCEDURE Radix50(VAR name: FileName;
                  VAR name50: Rad50name);
PROCEDURE SetBlock(f: FILE; VAR fn: FileName; func,
                   l: CARDINAL; VAR reply: INTEGER);
  (* func: код функции: 0 = Delete; 1 = Lookup;
                        2 = Create
     l:    длина файла для Create
     reply: >=0 = выполнено, длина файла
             <0 = ошибка
             -1 = канал занят
             -2 = файл не найден/нет памяти  *)
PROCEDURE TransmitBlock(f: FILE; func: CARDINAL;
                        PtrToBuf: ADRESS; blknr,
                        wcount:WORD;
                        VAR reply: INTEGER);
  (* func:   10B = чтение; 11B = запись
     reply:  >=0 = количество переданных слов
              <0 = ошибка
              -1 = сбой оборудования
              -2 = канал не открыт  *)
END Files.
```


16.4. Ввод и вывод с терминала

Этот модуль импортирует процедуры, которые читают и записывают знаки с операторской консоли. Процедура SetMode разрешает производить чтение в соответствии с различными режимами работы.

```
DEFINITION MODULE TTIO;
  EXPORT QUALIFIED Read, ReadAgain, Write, SetMode,
                   Line, WriteLn, WriteString;
  TYPE Line = ARRAY [0..80] OF CHAR;
  PROCEDURE Read(VAR ch: CHAR);
  PROCEDURE ReadAgain;
    (* вернуть последний знак назад в буфер, чтобы его можно было прочесть еще раз. Возврат только на один знак *)
  PROCEDURE Write(ch: CHAR);
  PROCEDURE SetMode(m: CARDINAL; cc: BOOLEAN);
    * устанавливаются режимы чтения:
m=0: нет эхо-печати, нет обработки специальных знаков,
     возврат из процедуры после ввода знака.
m=1: как m=0, но если знак не введен, то возврат кода 0C.
m=2: эхо-печать, работа с буфером до появления <ВК>,
     автоматическая обработка backspase, разрешен ввод
     командных файлов.
cc=TRUE:  ^C прерывает программу в ФОДОС-2
cc=FALSE: запрещено ^C прерывать программу,
          ^C является обычным символом для
          программы. *)
  PROCEDURE WriteLn;
  PROCEDURE WriteString(s: Line);
END TTIO.
```

16.5. Управление памятью

Экспортируемые процедуры ALLOCATE и DEALLOCATE предназначены для предоставления и освобождения памяти при динамическом распределении переменных. Распределяемое пространство определяется указателем «P»; его размер — параметром SIZE.

Пусть имеется описание:

```
TYPE T=...;
  VAR P: POINTER TO T;
```

тогда операторы NEW(P) и DISPOSE(P) переведутся компилятором в вызовы:

```
ALLOCATE(P,TSIZE(T));
DEALLOCATE(P,TSIZE(T)).
```

Поэтому, если NEW и DISPOSE встречаются в программе, процедуры ALLOCATE и DEALLOCATE должны быть импортированы либо описаны в самом модуле. Обычно они импортируются из стандартного модуля Storage, однако, возможен импорт их из другого модуля, написанного программистом.

```
DEFINITION MODULE Storage;
FROM SYSTEM IMPORT ADDRESS;
EXPORT QUALIFIED ALLOCATE, DEALLOCATE, SetMode;

  PROCEDURE ALLOCATE(VAR p: ADDRESS; size: CARDINAL);
  PROCEDURE DEALLOCATE(VAR p:ADDRESS; size:CARDINAL);
  PROCEDURE SetMode(m: CARDINAL);
    (* m: 1 = ALLOCATE прекращает работу, если не
                       достаточно памяти
          2 = ALLOCATE дает значение NIL, если не
                       достаточно памяти  *)
END Storage.
```

16.6. Загрузчик

Этот модуль экспортирует процедуру Call, которая записывает в память и выполняет загрузочные модули, создаваемые компоновщиком. Физические адреса определяются при компоновке. При этом используется стековая схема распределения памяти. Переменная FirstFree указывает адрес вершины стека программ.

```
DEFINITION MODULE Loader;
  FROM SystemTypes IMPORT FileName, LoadResultType,
                          ErrorType;
  EXPORT QUALIFIED Call, FirstFree;
  VAR FirstFree: CARDINAL;  (* адрес первой свободной
                               ячейки после области
                               программы. Только для
                               чтения *)
  PROCEDURE Call(fn: FileName; VAR LoadRes:
                             LoadResultType;
                             VAR ExecutionRes: ErrorType);
      (* загружает и выполняет программу
         fn:            имя загружаемого файла
         LoadRes:       результат загрузки
         ExecutionRes:  «результат» выполнения *)
END Loader.
```

16.7. Планировщик процессов

В последние годы разработано несколько языков программирования с концепциями параллельного программирования.

Отметим CONCURRENT, PASCAL, MODULA, PEARL и PORTAL. Все они используют понятие последовательного процесса, дополненное возможностью синхронизации процессов. В языке MODULA-2 вместо этого введено понятие сопрограммы и средства передачи управления от одной программы к другой.

Модуль, описанный здесь, показывает, как процессы и синхронизирующие сигналы могут быть описаны в терминах сопрограмм и операторов передачи управления. Модуль ProcessScheduler экспортирует процедуру запуска процесса — StartProcess. В ее параметрах указывается имя процедуры, которая станет процессом и переменную (массив), являющуюся рабочим пространством процесса. Модуль экспортирует тип SIGNAL и связанные с ним операторы SEND и WAIT. Каждый вызов SEND(S) возбуждает только один процесс, ожидающий S.

```
DEFINITION MODULE ProcessScheduler;
FROM SYSTEM IMPORT ADDRESS;

  EXPORT QUALIFIED
    SIGNAL, StartProcess, SEND, WAIT,
    Awaited, InitSignal;
  TYPE SIGNAL; (* сигналы должны быть инициализированы
                  процедурой InitSignal *)
  PROCEDURE StartProcess(P: PROC; A: ADDRESS; n: CARDINAL);
   (* запуск P с рабочей областью A длины n *)

  PROCEDURE SEND(VAR s: SIGNAL);
   (* возобновляет первый процесс, ждущий s *)
  PROCEDURE WAIT(VAR s: SIGNAL);
  PROCEDURE Awaited(s: SIGNAL): BOOLEAN;
  PROCEDURE InitSignal(VAR s: SIGNAL);
   (* инициализация переменных типа SIGNAL *)
END ProcessScheduler.
```

Существенная разница между языками, упомянутыми выше и языком MODULA-2, состоит в том, что в этих языках процессы — это отдельные свойства языка. Поэтому управление процессами требует резидентного, встроенного механизма. MODULA-2 позволяет (и требует) программирования таких механизмов. Они могут быть простыми или сложными в зависимости от целей и ограничений. Выше представлено одно из возможных решений в форме модуля ProcessScheduler. Основной характеристикой этого модуля является простота и наглядность реализации. Он весьма близок по свойствам к резидентному механизму языка MODULA. Здесь отсутствуют лишь непланируемые передачи управления, т. е. прерывания, представленные в языке MODULA оператором DOIO. В мо-

дуле ProcessScheduler сигналы представлены очередями (связанными списками дескрипторов) процессов. Переменная типа SIGNAL — это указатель на начало очереди. SEND отвязывает первый элемент от очереди; WAIT вносит дескриптор в конец очереди. Дескрипторы всех процессов связаны в кольцо. Это необходимо для поиска первого готового процесса.

17. СИНТАКСИЧЕСКАЯ ТАБЛИЦА

Приведено формальное описание синтаксиса языка и синтаксический словарь со ссылками к синтаксической таблице.

```
 1 Синтаксис        =<предложение>.
 2 Предложение      =нетермсим «=» выражение «.».
 3 Выражение =терм<«\» терм>.
 4 Терм             =фактор <«  » фактор>.
 5 Фактор =термсим\  нетермсим\  «(» выражение «)».
 6                «[»выражение«]» \ «<»выражение «>».
 7 Идентификатор =буква <буква\цифра>.
 8 Число            =целое\вещественное.
 9 Целое =цифра<цифра> \восьмеричная_цифра
   <восьмеричная_цифра>
10 («B»\«C») \цифра<шестнадцат_цифра> «H».
11 Вещественное =цифра<цифра> «.» <цифра>[
   показатель_степени].
12 Показатель_степени=«E»[«+»\«—»]цифра<цифра>.
13 Шестнадцат_цифра=цифра «A» «B» «C» «D» «E» «F».
14 Цифра           =восьмеричная_цифра\  «8»\  «9».
15 Восьмеричная_цифра=«0»\«1»\ «2»\«3»\«4»
\«5»\«6» \«7».
16 Строка          = «'»<знак>«'»\'«'<знак>'»'.
17 Квалидент=идентификатор <«.»идентификатор>.
18 Описание_константы=идентификатор «=» конст_выражение.
19 Конст_выражение=простое_конст_выраж[отношение
20                          простое_конст_выраж]
21 Отношение=«=»\«#»\ «< >»\«<»\«>»
\«<=»\«>=» \IN.
22 Простое_конст_выраж=[+\—]конст_терм <операция_
23 сложения конст_терм>.
24 Операция_сложения=«+»\«—»\OR.
25 Конст_терм=конст_фактор<операция_умножения
   конст_фактор>.
```



```
26 Операция_умножения=«*»\«/»\DIV\MOD\«&».
27 Конст_фактор =квалидент\число\строка \множество
28   «(»конст_выражение«)»\NOTконст_фактор.
29 Множество=[квалидент]«<»[элемент<«,»элемент>]«>».
30 Элемент   =конст_выражение[«..»конст_выражение].
31 Описание_типа     = идентификатор «=» тип.
32 Тип=простой_тип\тип_массива\ тип_запись\
33         тип_множества\тип_указателя\тип_процедуры.
34 Простой_тип =квалидент\перечисление\тип_диапазон.
35 Перечисление    =«(» список идентиф «)».
36 Список_идентиф=идентификатор <«,» идентификатор>.
37 Тип_диапазон =«[»  конст_выражение..конст_выражение«]».
38 Тип_массива =ARRAY простой_тип <«,» простой_тип>
   OF тип.
39 Тип_запись       =RECORD список_полей END.
40 Список_полей     =поле <«;» поле>.
41 Поле =[список_идент «:» тип\
42          CASE [идентификатор «:»] квалидент OF
43          вариант<«\»вариант >[ ELSE список_полей]
            END.
44 Вариант         =список_меток «:» список_полей.
45 Список_меток          =метка<«,» метка>.
46 Метка          =конст_выражение[«..»конст_выражение].
47 Тип_множества        =SET OF простой_тип.
48 Тип_указателя=POINTER TO тип.
49 Тип_процедуры =PROCEDURE [список_форм_пар].
50 Список_форм_пар=«(» [[VAR] формальный тип
51      «,» [VAR] формальный тип>] «)» [«:» квалидент].
52 Описание_переменных=список_идентификаторов «:» тип.
53 Обозначение      =квалидент <«,»идентификатор\
54          «[» список_выражений «]»\«^»>.
55 Список_выражений=выражение <«,» выражение >.
56 Выражение=простое_выражение[отношение простое_выражение].
57 Простое_выражение=[«+»\«—»]терм <операция_сложения терм>.
58 Терм          =фактор <операция_умножения фактор).
```

```
59 Фактор =число\строка\множество \обозначение
60 [фактич_параметры] \«(»выражение«)»\NOT фактор.
61 Фактич_параметры=«(»[список_выражений]«)».
62 Оператор =[присвоение\вызов_процедуры\оператор_IF \
63 оператор_CASE\оператор_WHILE\оператор_REPEAT \
64    оператор_LOOP\оператор_WITH\
65    EXIT\RETURN [выражение] ].
66 Присвоение          =обозначение «:=» выражение.
67 Вызов_процедуры=обозначение[фактич_параметры].
68 Посл_операторов=оператор <«;» оператор>.
69 Оператор_ IF=IF выражение THEN посл_операторов
70 <ELSIF выражение THEN посл_операторов>
71 [ELSE посл_операторов] END.
72 Оператор_CASE     =CASE выражение OF CASE <«\»
   CASE>
73 [ELSE посл_операторов] END.
74 CASE              =список_меток «:» посл_операторов.
75 Оператор_ WHILE=WHILE выражение DO посл_опера-
   торов END.
76 Оператор REPEAT=REPEAT посл_операторов UNTIL
   выражение.
77 Оператор FOR=FOR идентификатор «:=» выражение
     TO выражение
78 [BY конст_выражение] DO посл_операторов END.
79 Оператор LOOP     =LOOP посл_операторов END.
80 Оператор WITH     =WITH обозначение DO посл_опе-
   раторов END.
81 Описание_процедуры = заголовок_процедуры «;» блок
   идентификатор.
82 Заголовок_процедуры = PROCEDURE идентификатор
   [формальные параметры].
83 Блок          =<описание> BEGIN посл_операторов END.
84 Описание =CONST <описание_константы«;»>\
85 TYPE <описание_типа«;»>\
86 VAR <описание_переменных «;»>\
87 (описание_процедуры ;»>\
88 <описание_модуля «;»>.
89 Формальные_параметры=«(»[описание_форм_пар
90 <«;»описание_форм_пар>]«)» [«:»квалидент].
91 Описание_форм_пар = [VAR] список_идентификаторов
                          «:» формальный_тип.
```



```
92 Формальный_тип = [ARRAY OF] квалидент.
93 Описание_модуля=MODULE имя_модуля[приоритет]
     «;» [импорт]
94 [экспорт] блок имя_модуля.
95 Имя_модуля        = идентификатор.
96 Приоритет=«[»целое«]».
97 Экспорт =EXPORT [QUALIFIED] список_идентифика-
   торов «;».
98 Импорт = [FROM имя_модуля] IMPORT список_иден-
   тификаторов«;».
99 Модуль_определений = DEFINITION MODULE имя_мо-
   дуля «;» <импорт>
100 <экспорт> <описание> END имя_модуля.
101 Описание =CONST <описание_константы «;»>\
102          TYPE <идентификатор [«=» тип ] «;»>\
103       VAR <описание_переменных>\
104       <заголовок_процедуры «;»>.
105 Программный_модуль = MODULE имя_модуля[PRIO-
   RITET] «;» <импорт>
106 блок имя_модуля «.».
107 Единица_компиляции=модуль_определений\
         [IMPLEMENTATION[
108       программный_модуль «.».
```

MODULA-2. РУКОВОДСТВО ПРОГРАММИСТА

1. СИСТЕМА MODULA-2

Система MODULA-2 предназначена для разработки программ на языке программирования MODULA-2. Она состоит из ядра, которое является резидентной частью системы, компилятора, компоновщика, интерпретатора команд и отладчика.

После загрузки системы, управление передается интерпретатору команд. Он выводит приглашение на терминал и ждет ввода с клавиатуры имени файла программы, подлежащей выполнению. Получив имя файла, интерпретатор команд вызывает загрузчик, входящий в состав ядра, который загружает указанный файл и передает на него управление. По завершении программы управление возвращается в ядро и вновь загружается интерпретатор команд. Компилятор, компоновщик и отладчик написаны на языке MODULA-2. Они трактуются ядром также, как и любая пользовательская программа.

Дополнительная инфорррмация о реализации и использовании системы MODULA-2 содержится в описании языка MODULA-2, р. 16.

2. ТЕРМИНОЛОГИЯ И ПРИМЕРЫ

2.1. Терминология

Единица компиляции: модуль определений или программный модуль (см. синтаксис языка MODULA-2).
Модуль определений: часть отдельного модуля, специфицирующая экспортируемые объекты.
Программный модуль: часть отдельного или основного модуля, специфицирующая реализацию.
Исходный файл: входной файл компилятора (единица компиляции). Расширение по умолчанию MOD.
Файл символов: выходной файл компилятора с информацией

о таблице символов. Генерируется во время компиляции модуля определений. Расширение по умолчанию SYM.
Файл ссылок: выходной файл компилятора с информацией для отладчика. Генерируется во время компиляции программного модуля. Расширение по умолчанию REF.
Объектный файл: выходной файл компилятора. Содержит объектный код в формате системы MODULA-2. Расширение по умолчанию LNK.
Загрузочный файл: выходной файл компоновщика, содержащий загрузочный код в формате системы MODULA-2. Расширение по умолчанию LOD.
Файл MAP: выходной файл компоновщика с информацией о распределении памяти. Расширение по умолчанию MAP.

2.2. Примеры

В описании используются следующие примеры:

```
MODULA PROG1;
  . . .
END PROG1.
MODULE PROG2;
BEGIN
   A:=2
END PROG2.
DEFINITION MODULE PROG3;
   EXPORT QUALIFIED . . .
  . . .
END PROG3.
IMPLEMENTATION MODULE PROG3;
  . . .
END PROG3.
DEFINITION MODULE PROG4;
   IMPORT SYSTEMTYPES;
   EXPORT QUALIFIED . . .
  . . .
EDN PROG4.
IMLEMENTATION MODULE PROG4;
   IMPORT SYSTEMTYPES,LOADER,EXCEPTIONS;
  . . .
MODULE PROG5;
   IMPORT PROG3,PROG4;
  . . .
END PROG5.
```

3. ВХОД И ВЫХОД

Прежде, чем начать работу, необходимо выполнить команду:

```
SET SL OFF
```

Ядро системы загружается и запускается по команде операционной системы ФОДОС-2:

```
.R MODULA
*
```

Сразу после этого загружается и выполняется интерпретатор команд. Он выводит на терминал звездочку «*», показывая тем самым свою готовность для получения имени файла. Расширение имени по умолчанию LOD. Этот файл должен быть предварительно сгенерирован компоновщиком. Для выхода из системы необходимо нажать клавиши <СУ> и <С>.

4. КОМПИЛЯЦИЯ

Для работы компилятора необходимо создать предварительно три рабочих файла на устройстве с логическим именем WRK:

```
WRK:IL1.M2C      (50 блоков)
WRK:IL2.M2C      (50 блоков)
WRK:ASCII.M2C    (20 блоков)
```

Для доступа к устройству WRK: необходимо выполнить назначение, например: ASS DK WRK

4.1. Компиляция программного модуля

Для вызова компилятора необходимо ввести имя файла COMP. После вывода строки «SOURCE FILE>» компилятор ждет ввода имени файла, содержащего единицу компиляции.

Устройство по умолчанию предполагается: DK.

Расширение по умолчанию предполагается: MOD.

```
*COMP
SOURCE FILE>PROG1 (* введено имя DK:PROG1.MOD*)
P1
P2                    (* индикация последовательности *)
P3                    (* активированных проходов      *)
P4                    (*          компилятора         *)
P5
END COMPILATION
*
```


Если компилятор обнаружил в программе синтаксические ошибки, то он завершает свою работу после третьего прохода и генерирует листинг с сообщениями об ошибках.

```
*COMP
SOURCE FILE> PROG2
P1
------ERROR
P2
P3
------ERROR
  LISTER
END COMPILATION
*
```

4.2. Компиляция модуля определений

Для модуля определений рекомендуется расширение имени файла DEF. Модуль определений должен компилироваться перед своей программной частью (модулем реализации). Результатом компиляции является файл символов.

```
*COMP
SOURCE FILE> PROG3.DEF    (* модуль определений *)
P1
P2
SYMFILE
LISTER
END COMPILATION
*
```

4.3. Файлы символов

В процессе компиляции модуля определений генерируется файл символов, содержащий информацию для компилятора о таблице символов. Она необходима компилятору в двух случаях:

1) при компиляции программной части модуля;
2)при компиляции других единиц, импортирующих объекты из этого отдельного модуля.

Если после имени файла, подлежащего компиляции, указан ключ Q (см. п. 4.5), то компилятор запрашивает имена необходимых ему файлов символов, иначе он берет их по умолчанию (первые шесть символов от имени модуля).

```
*COMP
SOURCE FILE> PROG3        (* программный модуль *)
P1
PROG3 : DK:PROG3.SYM
P2
P3
```

```
P4
P5
END COMPILATION
*
*COMP
SOURCE FILE> PROG4.DEF
P1
SYSTEMTYPES : SY:SYSTEM.SYM
P2
SYMFILE
LISTER
END COMPILATION
*
*COMP
SOURCE FILE> PROG5/QUERY
P1
PROG3> PROG3
PROG4> PROG4
P2
P3
P4
P5
END COMPILATION
*
```

4.4. Файлы, генерируемые компилятором

Компилятор генерирует несколько файлов. Все они получают такое же имя файла как и исходный файл, но со следующими расширениями:

для модуля определений:	листинг	*.LST
	файл символов	*.SYM
для программного модуля:	листинг	*.LST
	файл ссылок	*.REF
	объектный файл	*.LNK

4.5. Набор ключей компилятора

За именем исходного файла могут следовать ключи компилятора. Ниже приводится описание ключей:

QUERY : компилятор явно спрашивает имена необходимых файлов символов, принадлежащих модулям, чьи объекты импортирует компилируемая единица (см. раздел 10).
Q — сокращенная форма этого ключа.

NOQUERY : компилятор не спрашивает имена файлов символов. Они ищутся в соответствии с принятой стратегией по умолчанию (см. раздел 10).

LISTING : должен быть сгенерирован файл листинга.

NOLISTING : файл листинга не генерируется.

N — сокращенная форма этого ключа.

80 : генерация кода для ЭВМ ряда «ЭЛЕКТРОНИКА МС0507».

60 : генерация кода для «ЭЛЕКТРОНИКА-60».

100 : генерация кода для «ЭЛЕКТРОНИКА-100/25».

VERSION : компилятор выводит на терминал информацию о текущей версии (т. е. тип процессора и операционной системы).

V — сокращенная форма этого ключа.

Ключи NOQUERY и LISTING устанавливаются по умолчанию. Ключи по умолчанию в отношении типа процессора и операционной системы устанавливаются при генерации системы.

4.6. Ключи компилятора в тексте программы

Некоторые ключи могут быть размещены непосредственно в тексте программы в скобках комментария. Для этих ключей принят следующий синтаксис:

ключи = ключ [«,» ключ]
ключ = «$» буква переключатель
переключатель = «+» \ «—» \ «=».

Ключи должны стоять на первом месте в скобках комментария. Они не распознаются компилятором, если им предшествует другая информация.

Ключи:

S тест переполнения памяти.

T тест проверки выхода индекса за границы (массивов, оператора CASE).

Переключатели:

\+ код текста генерируется

— код теста не генерируется

= предыдущий переключатель становится действительным снова.

По умолчанию все переключатели устанавливаются в «+».

Пример:

```
MODULE X; (*$T+*)
   . . .                    (* код теста генерируется  *)
   . . .
   (*$T—*)
```

```
A[I]:=A[I+1];          (* код теста не генерируется *)
(*$T=*)
. . .
. . .                  (* код теста генерируется    *)
END X.
```

4.7. Код модуля

Для каждой единицы компиляции компилятор генерирует так называемый «Код модуля». Этот код является уникальным и необходимым для того, чтобы отличить несколько скомпилированных версий одного и того же модуля. Код модуля записывается в файл символов, объектный файл и загрузочный файл.

Для программного модуля свой код не генерируется. Он получает тот же код, что и соответствующий ему модуль определений. Коды импортируемых модулей также записываются в файлы символов и объектные файлы.

Любое несовпадение кодов, принадлежащих одному модулю вызовет сообщение об ошибке во время компиляции или компановки..

ВНИМАНИЕ: перекомпиляция модуля определений создаст новый файл символов с новым кодом модуля. В этом случае программный модуль и все другие модули, импортирующие объекты этого модуля, должны быть тоже перекомпилированы. Перекомпиляция программного модуля не изменяет кода модуля.

5. КОМПОНОВКА

Объектный код, получаемый компилятором (файл с расширением LNK), должен быть скомпонован перед выполнением с файлами, чьи объекты он импортирует. Полученный в результате кодовый файл (называемый загрузочным файлом с расширением LOD) может быть загружен для выполнения.

Для вызова компоновщика необходимо ввести имя файла LINK. После вывода строки «MASTER FILE>» компоновщик ждет ввода имени файла, содержащего основную программу.

```
*LINK
MASTER FILE> PROG1          (* DK:PROG1.LNK *)
```

Компоновщик генерирует два файла с таким же именем как файл основной программы, но со следующими расширениями:

LOD : загрузочный файл, содержащий загружаемый код.

MAP : карта с распределением адресов скомпонованных модулей (только если установлен ключ M).

Программа должна всегда компоноваться к некоторой базе. Информация об этой базе необходима компоновщику. Для получения этой информации компоновщик запрашивает имя файла, содержащего загрузочный модуль базы. Обычно базой является резидентная часть ядра, т. е. файл SY:MODULA.M2S. Он выбирается компоновщиком по умолчанию как базовый, если другой файл не указан (см. ключ B).

Все отдельные модули, импортируемые основным модулем (или другими, компонуемыми к нему модулями), должны быть скомпонованы. Для компоновки необходимого модуля компоновщику требуется соответствующий объектный файл. Для поиска этого файла может быть использована стратегия по умолчанию или задан ключ Q.

Ключи компоновщика вводятся с клавиатуры. Они должны следовать за именем основного файла (см. раздел 9).

Компоновщик имеет следующий набор ключей:

B : компоновщик спрашивает имя файла базы

Q : компоновщик спрашивает имена объектных файлов, содержащих импортируемые модули (см. раздел 10). Если ключ Q не задан, то файлы ищутся в соответствии со стратегией по умолчанию (см. раздел 10).

M : генерируется файл с картой распределения адресов.

V : компоновщик выводит на терминал информацию о текущей версии.

S : ключ говорит о том, что должна генерироваться система, т. е. не должно быть привязки ни к какой базе.

A : используется только в совокупности с ключом S. Компоновщик спрашивает стартовый (самый младший) адрес вновь генерируемой системы. Адресное пространство между 400B и стартовым адресом резервируется для ядра системы MODULA-2.

Если ключ A не установлен, то стартовый адрес предполагается равным значению по умолчанию (5500B).

```
*LINK                       (* базой является ядро    *)
MASTER FILE> PROG4/Q        (* импортирующие           *)
END LINKAGE                 (* модули уже скомпонова-  *)
*                           (* ны в резидентной части  *)
*LINK
MASTER FILE> PROG5/B/Q/M
BASE FILE> PROG4
```

```
LINK FILES :
PROG3> PROG3
END LINKAGE
*
```

6. ЗАПУСК ПРОГРАММЫ

Скомпонованная программа готова для вызова (т. е. загрузки и выполнения). Существует две разных возможности вызова программы:
1) передать имя файла интерпретатору команд;
2) активировать процедуру CALL, экспортируемую из модуля LOADER.

6.1. Вызов через интерпретатор команд

Программа, скомпонованная с резидентной частью ядра (файл MODULA.M2S), вызывается посредством передачи ее имени файла интерпретатору команд. После вывода на экран звездочки «*» интерпретатор команд готов получить это имя.

Устройства по умолчанию: DK (первое устройство по умолчанию).
SY (второе устройство по умолчанию).

Если устройство не специфицировано, то файл ищется в первую очередь на DK. Если его там не оказалось, то поиск продолжается на устройстве SY.

Расширение по умолчанию : LOD

Если во время загрузки или выполнения имела место ошибка, то интерпретатор команд выводит на экран сообщение об ошибке и в файл SY:DUMP.COR будет записано содержимое памяти. Этот файл может быть проанализирован с помощью отладчика.

```
*PROG1 загружается и выполняется программа PROG1.LOD
. . .
*PROG5
------ WRONG LOAD KEY программа не скомпонована с ядром.
```

6.2. Организация оверлеев

В системе MODULA-2 реализована так называемая оверлейная организация. Это простой инструмент для наложения (перекрытия) частей программы. Оверлейная часть является загрузочным файлом, полученным компоновщиком. Она состоит из одного или более скомпонованных модулей. Оверлейная часть загружается и выполняется только в том случае, когда она вызвана из базовой части, с которой она была скомпонована (файл базы). Это простое правило позволяет иметь наложение программ на нескольких уровнях.

В системе MODULA-2 ядро является первым уровнем наложения. Все программы, запускаемые через интерпретатор команд, выполняются на втором уровне наложения.

6.3. Вызов из программы

Если программа P2 не скомпонована с ядром, то она должна быть вызвана непосредственно из базовой части P1, с которой она была скомпонована, т. е. должна быть активирована процедура CALL, экспортируемая из модуля LOADER (см. определение модуля EXCEPTIONS).

Пример: фрагмент из модуля PROG4.

```
VAR LOADRES : SYSTEMTYPES. LOADRESULTTYPE;
    EXERROR : SYSTEMTYPES. ERRORTYPE;
. . .
LOADER.CALL («DK PROG5 LOD», LOADRES, EXERROR);
IF EXERROR <> SYSTEMTYPES. NORMALRETURN THEN
(* отменить выполнение *)
EXCEPTIONS. RAISE (SYSTEMTYPES. PROPAGATE);
ELSE
. . .
```

6.4. Получение программ в загрузочном формате ФОДОС-2

Система MODULA-2 позволяет получать программы, которые запускаются непосредственно из ФОДОС-2.

Для этого должен быть указан ключ S при компоновке этой программы. Затем необходимо запустить программу SYSGEN. Она объединяет файл, полученный во время компоновки с файлом RTS.M2S и преобразует их в загрузочный

файл в формате ФОДОС-2. Ключ S указывает, что данная программа не должна быть связана с базой. Для указания начального адреса, с которого должен начинаться код, может быть указан ключ A (см. раздел 6).

```
*LINK
MASTER FILE> PROG4/S
LINKING SYSTEM
LINK FILES:
SYSTEMTYPES: SY: SYSTEM.LNK
LOADER: SY: LOADER.LNK
EXCEPTIONS: SY: EXCEPT.LNK
END LINKAGE
*
```

Программа SYSGEN запрашивает начальное значение указателя стека (восьмеричное число). Если напечатать ноль, то воспринимается значение по умолчанию 131000B. Затем требуется ввести имя файла, который она будет преобразовывать, объединив его с файлом RTS.M2S. В результате будет получен загрузочный файл с тем же именем и с расширением SAV.

```
.R SYSGEN
STACK START ADDRESS (O FOR DEFAULT VALUE 131000B) : 0
LOAD FILE: PROG4          (* файл DK: PROG4.LOD *)
END SYSTEM GENERATION
```

Теперь сгенерированная программа может быть выполнена под управлением ФОДОС-2.

```
.R PROG4
```

ВНИМАНИЕ: будьте осторожны! Выполнение программы, преобразованной в загрузочный формат ФОДОС-2, является опасным, т. к. все возможности ядра по обработке ошибочных ситуаций здесь отсутствуют. Любая ошибка в программе будет вызывать прерывание, которое должно быть получено и обработано ею самой.

6.5. Получение программы в формате LDA

С помощью программы SYSLDA, загрузочный файл в формате системы MODULA-2 может быть преобразован в файл в формате LDA. Для этого при компоновке необходимо указать ключ S и, возможно, A, также как и при получении программы в загрузочном формате ФОДОС-2.

Программа SYSLDA спрашивает имя загрузочного файла и начальное значение указателя стека генерируемой программы.

7. ОТЛАДЧИК

Если во время выполнения программы имела место ошибка и программа завершилась аварийно, то система MODULA-2 записывает текущее состояние всей памяти в файл дампа SY:DUMP.COR. Этот файл затем может анализироваться отладчиком.

Для запуска отладчика введите его имя файла DEBUG. Он запрашивает имя загрузочного файла программы, которая завершилась аварийно. Для работы отладчику необходимы также файлы ссылок и файлы листингов, принадлежащие всем отдельным модулям, скомпонованным в этой программе. Если эти файлы не найдены, то «окно» D и «окно» T будут «зашторены». Получив имя загрузочного файла, отладчик генерирует цепочку вызова процедур и дальше ждет ввода команды.

```
*DEBUG
MODULA-2 DEBUGGER
-----------------
FILE NAME:
```

7.1. «Окна» отладчика

Отладчик предлагает возможность обзора аварийно завершившейся программы через четыре различных «окна», любое из которых может быть выбрано альтернативно:

P окно: показывает цепочку активированных процедур в момент аварии.

D окно: показывает значение переменных до момента аварии, принадлежащих выбранной процедуре или отдельному модулю.

T окно: показывает исходный текст процедуры или отдельного модуля.

C окно: представляет возможность посмотреть содержимое файла дампа памяти.

Для анализа программы с помощью отладчика необходимо через P «окно» выбрать один объект цепочки процедур. Затем через другие «окна» можно посмотреть на данные и текст, принадлежащие этому объекту. Чтобы выбрать другой объект, надо вернуться в P «окно».

7.2. Команды отладчика

Команды переключения «окон»:

P : переключение к P окну
D : переключение к D окну
T : переключение к T окну
C : переключение к C окну
Q : выход из отладчика.

Для каждого «окна» существует свой набор команд:

P окно : + сместить указатель вверх
— сместить указатель вниз
? показать действительную позицию в цепочке

D окно : + сместить указатель в пределах того же уровня
— сместить указатель в пределах того же уровня
F сместить указатель на уровень «Отца» (вверх)
S сместить указатель на уровень «Сына» (вниз)
M отладчик спрашивает имя отдельного модуля и выводит его переменные
? Показать действительную позицию в дереве данных

T окно : + вывести предыдущие строки
— вывести следующие строки
M отладчик спрашивает имя отдельного модуля и выводит его исходный текст

C окно : = отладчик спрашивает адрес (восьмеричный или десятичный) и показывает содержимое этого адреса, а также его окружение.

После ввода команд «M» и «=» пользователь должен ввести дальнейшую информацию, т. е. имя модуля или восьмеричный адрес соответственно. Ошибочные символы могут быть удалены клавишей <ЗБ>. Сообщения системы не изменяют состояния отладчика, т. е. можно продолжать работу. Неправильные команды система игнорирует.

8. ИМЕНА ФАЙЛОВ И КЛЮЧИ

Интерпретатор команд, компилятор и компоновщик используют одни и те же модули для ввода с клавиатуры имен файлов (в формате ФОДОС-2) и ключей.
Ниже приведен синтаксис спецификаций файла:
ввод=имя файла [ключ] (<ВК>\« »)
имя файла=[[идент] «:»] [идент] [«.» [идент]]
идент=(буква\цифра) [буква\цифра]
ключ=«/»идент.

Завершением ввода спецификации файла с клавиатуры является возврат каретки или пробел. Ввод имени файла также может быть завершен следующим образом:

<СУ>/U отмена вводной строки
<ESC> нет файла.

9. СТРАТЕГИЯ ПОИСКА

Для поиска файлов, необходимых компилятору и компоновщику, существуют две стратегии: стратегия по умолчанию и стратегия по запросу.

Имя файла по умолчанию генерируется из имени модуля. Если имя модуля больше чем 6 символов, то для имени файла берутся первые 6 символов.

9.1. Стратегия по умолчанию

Имена всех необходимых файлов вырабатываются по умолчанию методом описанным выше. Сначала файл ищется на устройстве DK. Если поиск оказался неудачным, то файл ищется на устройстве SY.

По каждому файлу выводится сообщение о том, где он был найден, или, что он вообще не найден. В последнем случае пользователь не имеет возможности ввести другое имя файла.

9.2. Стратегия по запросу

Для каждого искомого файла пользователь вводит имя файла (устройство по умолчанию DK). Если поиск не удался, то выводится сообщение об ошибке и пользователь может ввести другое имя файла. Если введен только символ возврата каретки, то получение имени файла и его поиск ведутся в соответствии со стратегией по умолчанию.

Запрос на ввод имени файла повторяется до тех пор пока файл не будет найден или пока не будет введен символ <ESC>. Ввод символа <ESC> означает, что такого файла нет.

10. МОДУЛЬ SYSTEM

Модуль SYSTEM содержит некоторые дополнительные возможности системы MODULA-2. Большинство из них зависят от операционной системы и процессора.

Некоторые процедуры этого модуля необходимы для выполнения так называемых «низко-уровневых» операций (таких как обращение к операционной системе), другие процедуры предоставляют базовые средства для управления подпрограммой. Модуль SYSTEM непосредственно известен компилятору, так как объекты, экспортируемые им, подчиняются специальным правилам. Объекты, импортируемые из модуля SYSTEM, не нуждаются в соответствующем им файле символов для этого модуля. Более подробная информация об

этом содержится в описании языка MODULA-2 (разделы 12, 13).
Объекты, экспортируемые модулем SYSTEM:
типы:

```
        WORD
        ADDRESS
        PROCESS
    процедуры :
      NEWPROCESS(P:PROC; A:ADDRESS; N:CARDINAL; VAR P1:PROCESS)
      TRANSFER(VAR P1, P2: PROCESS)
      IOTRANSFER(VAR P1, P2: PROCESS; VA: CARDINAL)
      CALL(CODE: CARDINAL; REGO:WORD)
      CALL(CODE: CARDINAL; REGO:WORD; VAR CARRY: BOOLEAN)
      CALL(CODE: CARDINAL; REGO: WORD; VAR CARRY: BOOLEAN;
           TOPOFSTACK: WORD)
      CALL(CODE: CARDINAL; REGO: WORD; VAR CARRY: BOOLEAN;
                TOPOFSTACK, ... , TOPOFSTACK: WORD)
      LISTEN
      SYSRESET
    функция :
      REGISTER(NUM: CARDINAL): CARDINAL
```

11. СООБЩЕНИЯ КОМПИЛЯТОРА ОБ ОШИБКАХ

0 : Недопустимый символ
1 :
2 : Недопустимое значение константы
3 : Открытый комментарий в конце файла
4 : Конец строки символов не на этой строчке
5 : Слишком много ошибок
6 : Строка символов слишком длинная
7 : Слишком много идентификаторов (переполнена таблица идентификаторов)
8 : Слишком много идентификаторов (переполнена ХЭШ таблица)
20 : Ожидался идентификатор
21 : Ожидалась константа типа INTEGER
22 : Ожидался символ ']'
23 : Ожидался символ ';'
24 : Имя блока в конце неверно
25 : Ошибка в блоке
26 : Ожидался символ ':='
27 : Ошибка в выражении
28 : Ожидался символ 'THEN'
29 : Ошибка в операторе LOOP
30 : Константа не должна быть типа CARDINAL
31 : Ошибка в операторе REPEAT
32 : Ожидался символ 'UNTIL'

33 : Ошибка в операторе WHILE
34 : Ожидался символ 'DO'
35 : Ошибка в операторе CASE
36 : Ожидался символ 'OF'
37 : Ожидался символ ':'
38 : Ожидался символ 'BEGIN'
39 : Ошибка в операторе WITH
40 : Ожидался символ 'END'
41 : Ожидался символ ')'
42 : Ошибка в константе
43 : Ожидался символ '='
44 : Ошибка в объявлении типа
45 :
46 : Ожидался символ 'MODULE'
47 : Ожидался символ 'QUALIFIED'
48 : Ошибка в множителе
49 : Ошибка в задании простого типа
50 :
51 : Ошибка в типе формального параметра
52 : Ошибка в последовательности операторов
53 : Ожидался символ '.'
54 : Экспорт на глобальном уровне невозможен
55 : Модуль определений не может иметь программной части
56 : Ожидался символ 'TO'
57 : Модуль определений не может содержать в себе другие модули
58 : Ожидался символ ']'
59 : Ожидался символ '..'
60 : Ошибка в операторе FOR
61 : Ожидался символ 'IMPORT'
70 : Идентификатор специфицирован дважды в списке импорта
71 : Идентификатор не экспортируется из квалифицированного модуля
72 : Идентификатор объявлен дважды
73 : Идентификатор не объявлен
74 : Тип не объявлен
75 : Идентификатор уже объявлен в окружении данного модуля
76 : Динамический массив не должен быть параметром по значению
77 :
78 : Значение абсолютного адреса должно быть типа CARDINAL

79 : Переполнение таблицы области действия в компиляторе
80 : Недопустимый приоритет
81 : Не найден соответствующий модуль определений
82 : Структура не может быть применена для реализации скрытого типа
83 : Описание процедуры отлично от ее определения
84 : Не все определенные процедуры или скрытые типы описаны
85 :
86 : Несовместимые версии символьных модулей
87 :
88 : Тип функции не является скалярным или основным типом
89 :
90 : Тип указателя-ссылки не объявлен
91 : Ожидался тип поля признака
92 : Несовместимый тип метки выбора
93 : Константа используется дважды
94 : Арифметическая ошибка в преобразовании константного выражения
95 : Диапазон задан некорректно
96 : Диапазон допустим только для скалярных типов
97 : Несовместимый тип элемента
98 : Значение элемента вышло за границы
99 : Ожидался идентификатор типа SET
100 :
101 : Необъявленный идентификатор в списке экспорта модуля
102 :
103 : Неверный класс идентификатора
104 : Имя такого модуля не найдено
105 : Ожидалось имя модуля
106 : Ожидался скалярный тип
107 : Множество слишком велико
108 : Тип не должен быть INTEGER или CARDINAL
109 : Ожидался скалярный тип или тип диапазона
110 : Значение варианта вышло за пределы
111 : Экспорт из программного модуля недопустим
120 : Непозволительное преобразование типа
121 : Этот тип не ожидался
122 : Ожидалась переменная
123 : Некорректная константа
124 : Процедура для замены не найдена

125 : Неудовлетворительные параметры замененной процедуры
126 : Константа типа SET вышла за пределы
127 : Ошибка в параметрах стандартной процедуры
128 : Несовместимость типов
129 : Ожидался тип идентификатора
130 : Тип невозможно индексировать
131 : Поле не принадлежит переменной типа RECORD
132 : Слишком много параметров
133 :
134 : Ссылка не на переменную
135 : Недопустимая подстановка параметра
136 : Ожидалась константа
137 : Ожидались параметры
138 : Ожидался тип BOOLEAN
139 : Ожидались скалярные типы
140 : Операция с несовместимым типом
141 :
142 : Несовместимый тип элемента
143 : Операнды несовместимого типа
144 : Селекторы недопустимы для процедур
145 : В выражении разрешен только вызов функции
146 : Стрелка не принадлежит к переменной типа POINTER
147 : Стандартной процедуре или функции невозможно присвоение
148 : Константа недопустима как вариант
149 : Ожидался тип SET
150 : Неверная подстановка для параметра типа WORD
151 : Оператор EXIT возможен только внутри оператора LOOP
152 : Оператор RETURN возможен только в процедуре
153 : Ожидалось выражение
154 : Ныражение недопустимо
155 : Ожидался тип функции
156 : Ожидалась константа типа INTEGER
157 : Ожидался вызов процедуры
158 : Идентификатор не экспортируется из квалифицированного модуля
300 : Индекс вышел за границы
301 : Деление на ноль
302 :
303 : Метка оператора CASE определена дважды
400 : Выражение слишком сложное (переполнение регистра)

401 : Выражение слишком сложное (переполнение таблицы кодов)
402 : Выражение слишком сложное (слишком длинный переход)
403 : Выражение слишком сложное (переполнение таблицы переходов)
404 : Слишком много глобальных имен, внешних имен и вызовов процедур
405 : Процедура слишком длинная (переполнение таблицы кодов)
990 : Слишком глубокое вложение операторов WITH
991 : Делитель типа CARDINAL слишком большой (>100000B)
992 : Переменная цикла не должна иметь размер байта (для шага # —1, 0, 1)
993 : Функции INC и DEC со вторым аргументом не реализованы для байтовых переменных
994 : Слишком глубокое вложение процедур
995 : Шаг оператора FOR слишком большой (>77777B)
996 : Метка оператора CASE слишком большая

СОДЕРЖАНИЕ

ФОРТРАН/ФОДОС-2. ОПИСАНИЕ ПРИМЕНЕНИЯ

Стр.

1. Назначение системы 3
2. Условия применения 5
2.1. Требования к техническим средствам 5
2.2. Требования к программным средствам 5
3. Описание системы 6
3.1. Транслятор с ФОРТРАНа 6
3.2. Библиотека ФОРТРАНА 6
3.3. Файл бесформатного ввода-вывода 7
3.4. Диалоговый отладчик 8
3.5. Контрольные задачи 8
3.6. Вспомогательный файл 8
4. Входные и выходные данные 8
ПРИЛОЖЕНИЕ 1. Внешние функции 9
ПРИЛОЖЕНИЕ 2. Текст файла FORTRA.HLP 13
Перечень ссылочных документов 15

ФОРТРАН/ФОДОС-2. ОПИСАНИЕ ЯЗЫКА

1. Общие сведения 16
2. Способ описания языка 17
3. Элементы языка 18
3.1. Структура языка 18
3.2. Алфавит языка 19
3.2.1. Цифры 19
3.2.2. Буквы 19
3.2.3. Буквенно-цифровые символы 19
3.2.4. Специальные символы 19
3.3. Строки 20
3.3.1. Комментарии 20
3.3.2. Строка отладки 20
3.3.3. Заключительная строка 20
3.3.4. Начальная строка 21
3.3.5. Строка-продолжение 21
3.4. Предложения 21
3.5. Метка предложения 21
3.6. Символические имена 21
3.7. Упорядоченность символов 22
4. Элементы данных 22
4.1. Связь данного с его типом 22
4.2. Свойства данных разных типов 22
4.2.1. Тип целый 22

4.2.2. Тип вещественный . . . 22
4.2.3. Тип двойной точности . . . 22
4.2.4. Тип комплексный . . . 22
4.2.5. Тип логический . . . 23
4.2.6. Тип текстовый . . . 23
4.2.7. Тип байтовый . . . 23
4.3. Имена данных и процедур . . . 23
4.3.1. Константы . . . 23
4.3.1.1. Целая константа . . . 23
4.3.1.2. Восьмеричная константа . . . 24
4.3.1.3. Вещественная константа . . . 24
4.3.1.4. Константа двойной точности . . . 25
4.3.1.5. Комплексная константа . . . 25
4.3.1.6. Логическая константа . . . 25
4.3.1.7. Текстовая константа . . . 25
4.3.2. Переменные . . . 27
4.3.3. Массивы . . . 27
4.3.3.1. Элемент массива . . . 27
4.3.3.2. Индекс . . . 27
4.3.3.3. Индексное выражение . . . 27
4.3.4. Процедуры . . . 27
4.4. Ссылка на функцию . . . 28
4.5. Правила типов для идентификаторов данных и процедур . . . 28
4.6. Формальные параметры . . . 28
5. Выражения . . . 29
5.1. Арифметические выражения . . . 29
5.2. Отношения . . . 32
5.3. Логические выражения . . . 32
5.4. Вычисления выражений . . . 33
6. Операторы . . . 34
6.1. Операторы присваивания . . . 34
6.1.1. Арифметический оператор присваивания . . . 34
6.1.2. Логический оператор присваивания . . . 35
6.1.3. Оператор предписания . . . 35
6.2. Операторы управления . . . 36
6.2.1. Операторы перехода . . . 36
6.2.1.1. Безусловный оператор перехода . . . 36
6.2.1.2. Оператор перехода по предписанию . . . 37
6.2.1.3. Вычисляемый оператор перехода . . . 37
6.2.2. Условный арифметический оператор . . . 38
6.2.3. Условный логический оператор . . . 38
6.2.4. Оператор вызова подпрограммы . . . 38
6.2.5. Оператор возврата . . . 39
6.2.6. Оператор цикла . . . 39
6.2.7. Оператор продолжения . . . 42
6.2.8. Оператор паузы . . . 43
6.2.9. Оператор останова . . . 43
7. Средства обмена данными . . . 43
7.1. Общая характеристика и назначение операторов в-в . . . 43
7.1.1. Устройства ввода-вывода . . . 45
7.1.2. Списки ввода-вывода . . . 45
7.2. Объявление формата . . . 46
7.2.1. Описатели полей формата . . . 46
7.2.1.1. Описатель I . . . 49
7.2.1.2. Описатель O . . . 49

7.2.1.3. Описатель F . . . 50
7.2.1.4. Описатель E . . . 51
7.2.1.5. Описатель D . . . 51
7.2.1.6. Описатель G . . . 52
7.2.1.7. Описатель L . . . 53
7.2.1.8. Описатель A . . . 53
7.2.1.9. Описатель H . . . 54
7.2.1.10. Описатель '...' . . . 55
7.2.1.11. Описатель X . . . 55
7.2.1.12. Описатель T . . . 55
7.2.1.13. Описатель Q . . . 56
7.2.1.14. Описатель : . . . 56
7.2.1.15. Описатель $ . . . 56
7.2.2. Разделители полей и записей . . . 56
7.2.3. Взаимодействие форматного управления со списком ввода-вывода . . . 57
7.2.4. Масштабный множитель . . . 59
7.2.5. Спецификация формата в массивах . . . 60
7.3. Основные операторы ввода-вывода . . . 60
7.3.1. Операторы ввода-вывода последовательного доступа . . . 60
7.3.1.1. Оператор форматного ввода . . . 60
7.3.1.2. Оператор форматного вывода . . . 61
7.3.1.3. Оператор форматного ввода с терминала . . . 62
7.3.1.4. Оператор форматного вывода на терминал . . . 62
7.3.1.5. Оператор форматного вывода на построчно-печатающее устройство . . . 62
7.3.1.6. Оператор бесформатного ввода . . . 63
7.3.1.7. Оператор бесформатного вывода . . . 63
7.3.1.8. Оператор ввода с преобразованием по списку . . . 64
7.3.1.9. Оператор вывода с преобразованием по списку . . . 65
7.3.1.10. Оператор ввода с терминала с преобразованием по списку 66
7.3.1.11. Оператор вывода на терминал с преобразованием по списку 66
7.3.1.12. Оператор вывода на построчно-печатающее устройство с преобразованием по списку . . . 67
7.3.2. Операторы ввода-вывода прямого доступа . . . 67
7.3.2.1. Оператор бесформатного ввода прямого доступа . . . 67
7.3.2.2. Оператор бесформатного вывода прямого доступа . . . 68
7.4. Вспомогательные операторы ввода-вывода . . . 68
7.4.1. Оператор перемотки . . . 68
7.4.2. Оператор сдвига назад . . . 68
7.4.3. Оператор разметки . . . 69
7.4.4. Оператор описания файлов . . . 69
7.4.5. Оператор поиска записи . . . 70
7.4.6. Оператор открытия файла . . . 70
7.4.7. Оператор закрытия файла . . . 73
7.5. Операторы передачи . . . 73
7.5.1. Оператор передачи с преобразованием данных в код КОИ-7 73
7.5.2. Оператор передачи с преобразованием данных во внутреннее представление . . . 74
7.6. Вывод форматных записей на печать . . . 75
8. Объявления . . . 75
8.1. Объявление спецификаций . . . 76
8.1.1. Описание массива . . . 76
8.1.1.1. Функция линеаризации массива и значение индекса . . . 76
8.1.1.2. Регулируемые размеры . . . 77

8.1.2. Описание виртуального массива 78
8.1.3. Объявление массивов 78
8.1.4. Объявление виртуальных массивов 78
8.1.5. Объявление общих объектов 78
8.1.6. Объявление эквивалентности 80
8.1.7. Объявление внешних имен 81
8.1.8. Объявление типа 82
8.1.8.1. Правила, определяющие запись текстовых данных в памяти 83
8.1.9. Неявное объявление типа 84
8.2. Объявление начальных данных 84
9. Процедуры и модули 85
9.1. Внутренние функции 86
9.1.1. Структура объявлений внутренних функций 86
9.1.2. Ссылки на внутренние функции 87
9.2. Внешние функции 87
9.2.1. Структура модулей-функций 87
9.2.2. Ссылки на внешние функции 88
9.2.3. Основные внешние функции 89
9.3. Подпрограммы 89
9.3.1. Структура модулей-подпрограмм 89
9.3.2. Ссылки на внешние подпрограммы 90
9.4. Модуль-блок данных 91
9.5. Головной модуль 92
10. Средства отладки программы 92
ПРИЛОЖЕНИЕ 1. Операции ФОРТРАНа 93
ПРИЛОЖЕНИЕ 2. Предложения языка ФОРТРАН 93
ПРИЛОЖЕНИЕ 3. Основные внешние функции 95
Перечень ссылочных документов 99

ТРАНСЛЯТОР С ФОРТРАНа

1. Назначение и условия применения 101
1.1. Назначение 101
1.2. Условия применения 101
1.2.1. Требования к техническим средствам 101
1.2.2. Требования к программным средствам 102
2. Обращение к программе 102
2.1. Вызов транслятора по команде RUN 102
2.2. Вызов транслятора по команде FORTRAN 104
3. Характеристики программы. Входные и выходные данные . 104
3.1. Программный модуль 105
3.2. Объектный модуль 106
3.3. Листинг трансляции 106
3.4. Режим трансляции 110
3.4.1. Управление полями листинга 112
3.4.2. Виды объектного кода 112
3.4.3. Векторизация массивов 114
3.4.4. Номера внутренней последовательности 116
3.5. Логические и физические устройства 117
3.6. Связь программ и подпрограмм 118
3.7. Получение программ в абсолютном формате 119
3.8. Основные внешние функции 120
3.9. Организация памяти во время выполнения программы . . . 121
3.9.1. Оверлеи 123

3.9.2. Оверлеи расширенной памяти 124
3.10. Виртуальные массивы 125
3.11. Буферизация ввода-вывода 127
4. Сообщения 128
4.1. Сообщения синтаксического и семантического контроля . . . 130
4.1.1. Сообщения, предупреждающие об ошибках 140
4.1.2. Сообщения о неустранимых ошибках 140
ПРИЛОЖЕНИЕ 1. Коды символов 142
ПРИЛОЖЕНИЕ 2. Форматы данных 145
Перечень ссылочных документов 149

БИБЛИОТЕКА ФОРТРАНа

1. Назначение и условия применения программы 150
2. Характеристики программы 150
2.1. Структура библиотеки ФОРТРАНа 150
2.2. Объектный код 151
2.3. Вспомогательные подпрограммы библиотеки ФОРТРАНа . . 153
2.3.1. ASSIGN 154
2.3.2. CLOSE 156
2.3.3. DATE 157
2.3.4. IDATE 157
2.3.5. EXIT 157
2.3.6. USEREX 157
2.3.7. RANDU 158
2.3.8. SETERR 158
2.3.9. ERRTST 159
2.3.10. ERRSNS 159
2.4. Программные секции 159
2.5. Трассировка ошибок 159
3. Обращение к программе 160
4. Входные и выходные данные 161
5. Сообщения 161
ПРИЛОЖЕНИЕ. Программные секции 173
Перечень ссылочных документов 175

ДИАЛОГОВЫЙ ОТЛАДЧИК

1. Назначение и условия применения программы 176
1.1. Назначение 176
1.2. Условия применения 176
1.2.1. Требования к техническим средствам 176
1.2.2. Требования к программным средствам 176
2. Способ описания программы 177
3. Характеристики программы 177
3.1. Команды 178
3.1.1. Спецификация ячейки 179
3.1.1.1. Смещение 179
3.1.1.2. Именованный адрес 180
3.1.1.3. Относительный адрес 180
3.1.1.4. Индексированный адрес 180
3.1.1.5. Указатель типа 180
3.2. Паузы 181

4. Обращение к программе 182
4.1. Формирование загрузочного модуля 182
4.2. Вызов программы 182
5. Входные и выходные данные 183
5.1. Команды управления отлаживаемой программой . . . 183
5.1.1. Команды пауз 183
5.1.1.1. Команда входной и операторной паузы 183
5.1.1.2. Команда пошаговой паузы 185
5.1.1.3. Команда паузы слежения 185
5.1.1.4. Команда отмены входной и операторной паузы . . . 186
5.1.2. Команда повторного запуска 186
5.1.3. Команда продолжения 187
5.1.4. Команда останова 187
5.2. Команда организации ввода-вывода 187
5.2.1. Команды связи 187
5.2.1.1. Команда связи с именем переменной 187
5.2.1.2. Команда связи с описанием массива 188
5.2.1.3. Команда отмены связи 189
5.2.2. Команда ввода 189
5.2.3. Команда вывода 190
5.3. Команды управления отладчиком 191
5.3.1. Команды макросредств 191
5.3.1.1. Команда MACRO 191
5.3.1.2. Команда безусловного перехода 192
5.3.2. Команда условного перехода 192
5.3.3. Команда печати текущего состояния отладчика . . . 193
6. Сообщения 194
Перечень ссылочных документов 196

КОНТРОЛЬНЫЕ ЗАДАЧИ

1. Контрольная задача 1 197
1.1. Назначение программы 197
1.2. Условия выполнения программы 197
1.2.1. Требования к техническим средствам 197
1.2.2. Требования к программным средствам 197
1.3. Выполнение программы 198
1.3.1. Установка носителей 198
1.3.2. Загрузка и запуск транслятора с ФОРТРАНа 198
1.3.3. Получение объектного модуля и листинга программы DEMO1 198
1.3.4. Получение загрузочного модуля программы DEMO1 . . . 199
1.3.5. Запуск программы DEMO1 199
1.4. Сообщения оператору 200
2. Контрольная задача 2 200
2.1. Назначение программы 200
2.2. Условия выполнения программы 200
2.2.1. Требования к техническим средствам 200
2.2.2. Требования к программным средствам 200
2.3. Выполнение программы 201
2.3.1. Установка носителей 201
2.3.2. Загрузка и запуск транслятора с ФОРТРАНа 201
2.3.3. Получение объектного модуля и листинга программы DEMO 202
2.3.4. Получение загрузочного модуля программы DEMO . . . 202
2.3.5. Запуск программы DEMO 202
2.4. Сообщения оператору 203
ПРИЛОЖЕНИЕ 1. Текст программы DEMO1.FOR 203

ПРИЛОЖЕНИЕ 2. Листинг программы DEMO1.FOR 204
ПРИЛОЖЕНИЕ 3. Текст программы DEMO.FOR 206
ПРИЛОЖЕНИЕ 4. Листинг программы DEMO.FOR 206
Перечень ссылочных документов 208

СИСТЕМА УПРАВЛЕНИЯ ГРАФИЧЕСКИМ ТЕРМИНАЛОМ. ГРАФИЧЕСКИЙ ПАКЕТ

1. Назначение, условия применения и характеристики пакета программ 209
2. Обращение к подпрограммам пакета, входные и выходные параметры 210
3. Сообщения 210
4. Инициализация и абсолютная графика 210
4.1. Инициализация. Подпрограмма INITT 210
4.2. Окончание. Подпрограмма FINITT 211
4.3. Вычерчивание абсолютных векторов в координатах экрана . . 211
4.3.1. Подпрограмма MOVABS 211
4.3.2. Подпрограмма DRWABS 211
4.3.3. Подпрограмма PNTABS 212
4.4. Вычерчивание линий по приращениям экранных координат . . 213
5. Виртуальная и экранная графика 214
5.1. Виртуальное окно 214
5.2. Видимая область в координатах пользователя (виртуальное окно) 215
5.3. Определение виртуального окна 216
5.3.1. Подпрограмма VWINDO 216
5.3.2. Подпрограмма DWINDO 217
5.4. Вычерчивание линий в координатах пользователя (виртуальных) по абсолютным значениям координат 217
5.5. Подпрограммы для относительных виртуальных координат . . 218
5.6. Экранное окно 219
5.6.1. Подпрограмма SWINDO 220
5.6.2. Подпрограмма TWINDO 220
5.7. Масштабирование и растягивание экранного окна 220
5.8. Отсечение чертежа в виртуальном пространстве 221
5.9. Взаимозаменяемость виртуальной и экранной графики . . . 222
5.10. Вычерчивание пунктирных линий 223
5.10.1. Подпрограмма DASHA 223
5.10.2. Подпрограмма DASHR 223
5.10.3. Подпрограмма DSHABS 225
5.10.4. Подпрограмма DSHREL 225
5.11. Спецификация пунктирной линии: параметр L 225
6. Служебные процедуры 227
6.1. Алфавитно-цифровой вывод 227
6.2. Вход в алфавитно-цифровой режим. Подпрограмма ANMODE 227
6.3. Вывод алфавитно-цифровых символов. Подпрограмма ANCHO 227
6.4. Вывод алфавитно-цифровых символов. Подпрограмма ANSTR 228
6.5. Обработка алфавитно-цифровых символов 228
6.5.1. Подпрограмма NEWLIN 228
6.5.2. Подпрограмма LINEF 228
6.5.3. Подпрограмма CARTN 228
6.5.4. Подпрограмма HOME 229
6.5.5. Подпрограмма NEWPAG 229

6.5.6. Подпрограмма BAKSP . . . 229
6.6. Использование экранного курсора . . . 229
6.6.1. Подпрограмма SCURSR . . . 229
6.6.2. Подпрограмма DCURSR . . . 230
6.7. Использование виртуального курсора. Подпрограмма VCURSR 230
6.8. Область состояния терминала . . . 232
6.8.1. Подпрограмма SVSTAT . . . 232
6.8.2. Подпрограмма RESTAT . . . 232
6.9. Масштабирование и поворот . . . 233
6.9.1. Установка масштабирования . . . 234
6.9.1.1. Масштабирование графического вывода Подпрограмма RSCALE . . . 234
6.9.2. Установка поворота . . . 234
6.9.2.1 Поворот графического вывода. Подпрограмма RROTAT . . 235
6.10. Подпрограмма RESET . . . 236
6.11. Подпрограмма RECOVR . . . 236
6.12. Подпрограмма VECMOD . . . 236
6.13. Подпрограмма DSHMOD . . . 236
6.14. Подпрограмма LVLCHT . . . 236
6.15. Различные служебные подпрограммы . . . 236
6.15.1. Подпрограмма ERASE . . . 237
6.15.2. Подпрограмма BELL . . . 237
6.15.3. Подпрограмма SEETW . . . 237
6.15.4. Подпрограмма SEEDW . . . 237
6.15.5. Подпрограмма SEEREL . . . 237
6.15.6. Подпрограмма SEETRN . . . 238
6.16. Перевод дюймов в экранные единицы. Функция KIN . . 238
6.17. Перевод сантиметров в экранные единицы. Функция KCM . 238
6.18. Измерение ширины символов. Функция LINWDT . . . 239
6.19. Измерение высоты строк. Функция LINHGT . . . 239
6.20 Табуляция и задание границ (полей) . . . 239
6.20.1. Установка таблицы табулятора. Подпрограмма TTBLSZ . 239
6.20.2. Установка табулятора. Подпрограмма SETTAB . . . 240
6.20.3. Замена табулятора. Подпрограмма RSTTAB . . . 240
6.20.4. Горизонтальная табуляция. Подпрограмма TABHOR . . 240
6.20.5. Вертикальная табуляция. Подпрограмма TABVER . . . 241
6.20.6. Задание границ. Подпрограмма SETMRG . . . 242
6.21. Вспомогательные подпрограммы управления выводом графической информации . . . 243
6.21.1. Очистка текущего экранного окна. Подпрограмма CLEAR 243
6.21.2. Установка типа рисуемой линии. Подпрограмма LNSTYL 243
6.21.3. Установка типа рисования. Подпрограмма SETMOD . . . 243
7. Преобразования систем координат . . . 244
7.1. Линейное преобразование. Подпрограмма LINTRN . . . 246
7.2. Логарифмическое преобразование. Подпрограмма LOGTRN . 247
7.3. Полярное преобразование. Подпрограмма POLTRN . . . 247
7.4. Вычерчивание сегментов с использованием полярного преобразования. Подпрограммы DRAWSA и DRAWSR . . . 248
7.5. Черчение пунктирных сегментов линий в полярной системе координат. Подпрограммы DASHSA и DASHSR . . . 248
7.6. Применение полярных преобразований . . . 249
8. Подпрограммы ввода/вывода . . . 255
8.1. Вывод . . . 256
8.1.1. Подпрограмма TOUTST . . . 256
8.1.2. Подпрограмма TOUTPT . . . 257

8.1.3. Подпрограмма ANCHO . . . 257
8.1.4. Подпрограмма ANSTR . . . 257
8.1.5. Подпрограмма AIOUT . . . 257
8.1.6. Подпрограмма AOUTST . . . 258
8.2. Ввод . . . 258
8.2.1. Подпрограмма TINSTR . . . 258
8.2.2. Подпрограмма TINPUT . . . 258
8.2.3. Подпрограмма AIIN . . . 259
8.2.4. Подпрограмма AINST . . . 259
8.3. Подпрограммы обслуживания ввода/вывода . . . 259
8.3.1. Проверка свободного места во входном буфере. Подпрограмма-функция LEFTIO . . . 259
8.3.2. Определение положения графического луча на экране. Подпрограмма SEELOC . . . 260
9. Расширенное использование системы управления терминалом . . 260
10. Общие (глобальные переменные) области состояния терминала 264
10.1. Область общих переменных . . . 264
10.2. Порядок описания COMMON области . . . 265
11. Инструкции по работе с пакетом . . . 269
11.1. Работа с библиотекой . . . 269
11.2. Трансляция, компоновка и запуск графических программ . . 269
12. Словарь терминов системы управления терминалом . . . 269

MODULA-2. ОПИСАНИЕ ЯЗЫКА

1. Введение . . . 274
2. Описание синтаксиса . . . 275
3. Базовые понятия языка . . . 276
4. Описание идентификаторов . . . 278
5. Описание констант . . . 279
6. Описание типов . . . 280
6.1. Основные типы . . . 280
6.2. Тип перечисление (ENUMERATIONS) . . . 281
6.3. Тип диапазона . . . 281
6.4. Тип массива (ARRAY) . . . 281
6.5. Тип записи (RECORD) . . . 282
6.6. Тип множества (SET) . . . 283
6.7. Тип указателя (POINTER) . . . 284
6.8. Тип процедуры (PROCEDURE) . . . 284
7. Описание переменных . . . 284
8. Выражения. . . . 285
8.1. Операнды . . . 285
8.2. Операции . . . 286
8.2.1. Арифметические операции . . . 286
8.2.2. Логические операции . . . 287
8.2.3. Операции с множествами . . . 287
8.2.4. Отношения . . . 287
9. Операторы . . . 288
9.1. Присваивания . . . 288
9.2. Вызовы процедур . . . 289
9.3. Последовательность операторов . . . 289
9.4. Оператор IF . . . 290
9.5. Оператор CASE . . . 290
9.6. Оператор WHILE . . . 290

9.7. Оператор REPEAT 291
9.8. Оператор FOR 291
9.9. Оператор цикла LOOP 292
9.10. Оператор WITH 292
9.11. Операторы RETURN и EXIT 293
10. Описание процедур 293
10.1. Формальные параметры 294
10.2. Стандартные процедуры 296
11. Модули 297
12. Системно-зависимые свойства языка 300
13. Процессы 301
13.1. Создание процессов и передача управления 301
13.2. Процессы для устройств и прерывания 302
14. Единица компиляции 304
15. Реализация и использование языка MODULA-2 305
16. Модули стандартных утилит 307
16.1 Ввод-вывод 307
16.2. Потоки 308
16.3. Файлы 310
16.4. Ввод и вывод с терминала 313
16.5 Управление памятью 313
16.6. Загрузчик 314
16.7. Планировщик процессов 314
17. Синтаксическая таблица 316

MODULA-2. РУКОВОДСТВО ОПЕРАТОРА

1. Система MODULA-2 320
2. Терминология и примеры 320
2.1. Терминология 320
2.2. Примеры 321
3. Вход и выход 322
4. Компиляция 322
4.1. Компиляция программного модуля 322
4.2. Компиляция модуля определений 323
4.3. Файлы символов 323
4.4. Файлы, генерируемые компилятором 324
4.5. Набор ключей компилятора 324
4.6. Ключи компилятора в тексте программы 325
4.7. Код модуля 326
5. Компоновка 326
6. Запуск программы 328
6.1. Вызов через интерпретатор команд 328
6.2. Организация оверлеев 328
6.3. Вызов из программы 329
6.4. Получение программ в загрузочном формате ФОДОС-2 329
6.5. Получение программы в формате LDA 330
7. Отладчик 331
7.1. «Окна» отладчика 331
7.2. Команды отладчика 331
8. Имена файлов и ключи 332
9. Стратегия поиска 333
9.1. Стратегия по умолчанию 333
9.2. Стратегия по запросу 333
10. Модуль SYSTEM 333
11. Сообщения компилятора об ошибке 334

Ответственный за выпуск **М. Г. Бойкова**
Редактор **Т. А. Савельева**
Корректор **В. Н. Лыткина**

Сдано в набор 15.12.89 г. Подписано в печать 2.03.90 г.
Формат 60×84¹/₁₆. Бумага писчая № 1. Гарнитура **«Литературная»**.
Печать высокая. Усл. печ. л. 20,45. Тираж 20 000 экз. Заказ 849.
Бесплатно.

Ленинградское отделение РППО «Союзбланкоиздат».
Великолукская городская типография управления издательств,
полиграфии и книжной торговли Псковского облисполкома,
182100, г. Великие Луки, ул. Полиграфистов, 78/12