Министерство высшего и среднего специального образования РСФСР

Куйбышевский ордена Трудового Красного Знамени авиационный институт имени С.П. Королева

Ю.Н. М а л и е в

ПРОГРАММИРОВАНИЕ НА ФОРТРАНЕ

Учебное пособие

Утверждено редакционно-издательским советом института I7.II.78 г.

Куйбышев I980

УДК 681.14

Программирование на ФОРТРАНЕ. М а л и е в Ю.Н. Учебное пособие. КуАИ, 1980 г.

В учебном пособии излагаются основные методы программирования задач на языке Базисный ФОРТРАН. Рассматриваются различные приемы программирования типовых задач, характерных для вычислительной техники. Приводятся особенности записи программ при решении задач в системе ДОС/ЕСЭВМ.

Пособие предназначено для студентов всех факультетов и специальностей, использующих средства вычислительной техники при выполнении курсовых заданий, курсовых и дипломных проектов, НИР и т.п. Пособием могут также воспользоваться преподаватели и научные работники для выполнения на ЭЦВМ трудоемких вычислений и расчетов.

Таблиц 2. Иллюстраций 6. Библиографий 7.

Издание второе, переработанное и дополненное.

Темплан 1980 г. поз. 1179.

Рецензенты: старший научный сотрудник ВО ИГ и РГИ АН СССР к.т.н. С а х а р о в Ю.П., заместитель начальника ИВЦ КПтИ к.т.н. К а л м ы к о в М.П.

П Р Е Д И С Л О В И Е

Алгоритмические языки находят все более широкое применение при решении на ЭВМ многих задач, связанных с различными областями науки и техники. Это обусловливает рост потребности в специалистах, владеющих определенными навыками программирования на таких языках. Данное учебное пособие является практическим курсом по программированию задач на алгоритмическом языке ФОРТРАН, весьма распространенном в настоящее время.

Язык ФОРТРАН (*FORTRAN - FORmula TRANSlation*) был разработан в 1954 году и предназначался для решения широкого класса инженерных и математических задач. За прошедшее время язык ФОРТРАН значительно расширил свои возможности. Существует несколько версий языка. Наиболее широко употребляются версии *BASIC FORTRAN* и *FORTRAN-IV*. Трансляторы с ФОРТРАНА имеют многие ЭВМ второго поколения, а также ЭВМ третьего поколения, в частности, все модели серии ЕС.

В пособии описаны основные средства базисной версии ФОРТРАНА, которые позволяют, однако, работать практически с любой версией ФОРТРАНА. Этих средств вполне достаточно для решения большинства задач вычислительного характера (желающие изучить более подробно все средства языка ФОРТРАН должны проработать литературу, в которой более обстоятельно рассматриваются соответствующие вопросы).

В I и II главах пособия приведены основные грамматические сведения о языке и структуре операторов. В III главе рассматриваются вопросы включения и использования подпрограмм. В V главе пособия даны указания по оформлению программ для решения на ЕС ЭВМ при использовании транслятора с базисного ФОРТРАНА, входящего в состав операционной системы ДОС/ЕС.

Материал книги иллюстрирован большим количеством примеров и упражнений и содержит рекомендации по использованию наиболее эффективных приемов программирования.

В заключение еще раз отметим, что поскольку базисный ФОРТРАН является подмножеством языка ФОРТРАН-IV, то программы, написанные на языке *BASIC FORTRAN*, без каких-либо изменений и дополнений могут обрабатываться транслятором языка *FORTRAN-IV*.

Глава I

ОСНОВНЫЕ ЭЛЕМЕНТЫ ФОРТРАНА

§I. АЛФАВИТ

ФОРТРАН построен на базе английского языка, поэтому он использует 26 заглавных букв латинского алфавита:
A, B, C, D, E, F, G, H, I, J, K, L, M,
N, O, P, Q, R, S, T, U, V, W, X, Y, Z.
Десять арабских цифр:
Ø, I, 2, 3, 4, 5, 6, 7, 8, 9.
Чтобы при перфорации программы отличить ноль от буквы О, его обычно перечеркивают наклонной чертой.

Специальные символы:

+	плюс
-	минус
*	звездочка
/	наклонная черта (деление)
=	равно (знак присвоения)
( )	скобки круглые
,	запятая
.	десятичная точка
/	апостроф
␣	пробел

Кроме того, в комментариях и текстах допускаются все символы (включая русский алфавит), имеющиеся во внешних устройствах (УПДК и АЦПУ).

Ключевые слова

Для построения программ на ФОРТРАНЕ используется определенный набор английских слов, каждое из которых имеет заданное функциональное назначение. По характеру действия их можно разделить на следующие группы.

Описатели:

INTEGER	целый
REAL	вещественный
DOUBLE PRECISION	двойная точность
FUNCTION	функция
SUBROUTINE	подпрограмма
EXTERNAL	внешний

Распределение памяти:

DIMENSION	размерность
COMMON	общий
EQUIVALENCE	эквивалентный

Управляющие:

GO TO	перейти к
IF	если
DO	выполнить
PROGRAM	программа
END	конец
CALL	вызвать
READ	читать

WRITE	писать
FORMAT	формат

С и г н а л и з а ц и и:

CONTINUE	продолжать
RETURN	вернуться

С л у ж е б н ы е:

PAUSE	останов
STOP	стоп

§2. ЧИСЛОВЫЕ КОНСТАНТЫ

Числовые константы могут быть следующих видов:

а) Целые (форма I);

б) Вещественные (форма *F* и *E*);

в) Удвоенной точности (форма *D*).

Целые константы применяются для записи только целых чисел и могут состоять не более чем из IO цифр. Знак может быть "+" или "-", причем знак "+" обычно не пишут.

Максимальное число: 2I47483647

Примеры: - 26
Ø
6I3754

Вещественные константы применяются для записи любых рациональных чисел, абсолютная величина которых находится в диапазоне $10^{-78} \div 10^{+75}$. Точность представления чисел составляет 7 десятичных разрядов. Для разделения целой и дробной части употребляется десятичная точка.

Применяются следующие формы записи вещественных констант (табл. I).

Числа с удвоенной точностью могут содержать до I7 цифр и записываются в общем виде, как
n.mD±k .

Таблица I

Общая форма записи		Запись на ФОРТРАНЕ	Обычная
Фиксир. запятая *Ф.F*	*n.m*	2.35	2,35
	n.	4231 4.	4231 4
	.m	.241	0,241
Плавающ. запятая *Ф.E*	*n.mE±K*	61.24EØ5	$61,24.10^{5}$
	n.E±K	247.E-Ø7	247.10^{-7}
	.mE±K	.255E12	$0,253.10^{12}$

Упражнение I.I. Написать следующие числа в виде вещественных констант в форме E:

1) -10^{-15}
2) 124
3) - 72,64
4) - 0,7
5) 0,000000724
6) 32.10^{+6}

Упражнение I.2. В приведенных ниже примерах объяснить ошибки в написании вещественных чисел:

1) 35.412E + 3.
2) 5.ØE
3) .15E + Ø14
4) 74E - Ø3

§3. ПРОСТАЯ ПЕРЕМЕННАЯ

Для наименования (идентификации) переменных применяются заглавные буквы латинского алфавита, а также любые комбинации из букв и цифр, которые должны начинаться с буквы и содержать не более шести символов.

Например: *X, TIME, MAX12, GAMMA, PRIM45, R25MIN.*

Описание переменных

В ФОРТРАНЕ существует два способа описания переменных (целых или вещественных).

В первом случае используется неявное определение типа пе-

ременной (принцип "умалчивания"). При этом целочисленные переменные обозначаются буквами из набора: *I, J, K, L, M, N* или словами, начинающимися с этих же букв: *MASSE, NOM32* . Для обозначения вещественных переменных используются остальные буквы алфавита. Никаких специальных описаний не приводят.

Во втором случае переменные определяются с помощью описаний (явное определение): *INTEGER, REAL, DOUBLE PRECISION.*

Например: *INTEGER X, MINI, Z15*
REAL MAX, N, A2X, TIME

Такие описания приводятся в начале программы.

§4. МАССИВЫ ПЕРЕМЕННЫХ

Кроме простых переменных в ФОРТРАНЕ могут использоваться переменные с индексами, которые позволяют представить большое количество величин одним общим наименованием. В этом случае любая переменная множества определяется одним, двумя или тремя индексами, которые пишутся в скобках после наименования массива.

Например: *A(2), M(3,7), Z(2,5,4).*

Наименования массивов выбираются по тем же правилам, что и для простых переменных.

В качестве индексов в выражениях можно использовать не только натуральные числа, но и переменные типа "целая", которым в ходе выполнения программы были присвоены положительные значения, а также выражения вида

$$\begin{cases} (I \pm m) \\ (n * I) \\ (n * I \pm m) \end{cases},$$

где m и n -целые числа, а I - целая переменная, причем m, n, I таковы, что значение каждого выражения, заключенного в скобки, должно быть положительным.

Например: *A(K); B(I-2); CR(3*N); DIV(2*I+3).*

Полная система переменных с индексами, объединенных общим наименованием, называется массивом, который обязательно должен быть описан в начале программы.

Описание массивов

Массивы определяются с помощью описателя *DIMENSION*, который указывает их размерность и величину.

Например: *DIMENSION A(...), B(...), C(...), ...,*
где *A, B, C* – наименования массивов, а в круглых скобках записываются одно, два или три натуральных числа. Количество этих чисел определяет размерность массива, а их величины показывают максимальное значение каждого индекса для тех переменных с индексами, которые являются элементами соответствующего массива.

Например: *DIMENSION A(4), B(3,2)*
означает, что массив *A* состоит из четырех элементов:
A(1), A(2), A(3), A(4),
а массив *B* состоит из шести элементов, расположенных в трех строках по два элемента в каждой:

B (I,I)	*B* (I,2)
B (2,I)	*B* (2,2)
B (3,I)	*B* (3,2).

Расположение массивов в памяти

Массивы чисел записываются в памяти таким образом, что значение первого (левого) индекса изменяется наиболее быстро.

Например, двухмерный массив M(3,3) будет располагаться в памяти в следующем порядке:

M(1,1) M(2,1) M(3,1) M(1,2) M(2,2) M(3,2) M(1,3) M(2,3) M(3,3).

Таким образом, двухмерные массивы размещаются в памяти по столбцам.

Элементарные математические функции

Для вычисления некоторых элементарных математических функций, например таких, как тригонометрические функции, логарифм, экспонента и другие, записывается условное наименование соответствующей функции и ее аргумент, заключенный в круглые скобки:

SIN(X)	$\sin x$,	x в радианах
COS(X)	$\cos x$,	x в радианах
TANH(X)	$th\ x$,	x в радианах
ATAN(X)	$Arc\,tg\,x$	
SQRT(X)	$\sqrt{x}$, $x \geq 0$	
ABS(X)	$\lvert x \rvert$	
ALOG(X)	$\ln x$	
ALOG10(X)	$lg\ x$	
EXP(X)	e^x	

Остальные тригонометрические функции можно получить либо косвенным путем, например: $arc\cos x = arc\,tg\,\frac{\sqrt{1-x^2}}{x}$, либо переходом к ФОРТРАНУ IУ, в котором имеются дополнительно следующие функции:
TAN(X), SINH(X), ARSIN(X),
COTAN(X), COSH(X), ARCOS(X).

Отметим, что аргумент X может быть любым арифметическим выражением, численное значение которого является допустимым для данной функции, например: SIN(A*X**5+B).

Для вычислений с удвоенной точностью имеется набор тех же функций, к наименованию которых добавляется буква D. Например, DSIN(X) - вычисление синуса с удвоенной точностью.

§5. АРИФМЕТИЧЕСКИЕ ВЫРАЖЕНИЯ

Над числовыми константами, переменными величинами и функциями можно производить обычные арифметические операции, каждая из которых обозначается особым символом:

Сложение + Умножение *
Вычитание - Деление /
Возведение в степень ** .

Арифметическими выражениями называются:

а) числовые константы,
например, 25; -.126; 3.7E-4;

б) переменные величины,
например, X; NOR; U18MK;

в) обращения к элементарным математическим функциям, например, *SIN(X); EXP(Y);*

г) обращение к подпрограммам типа *FUNCTION;*

д) числовые константы, переменные величины и функции, объединенные знаками арифметических операций и круглыми скобками,
например, *A+(B2*X-25.)**2-SIGMA/4.6*

При построении арифметических выражений надо соблюдать следующие правила.

I. Арифметические выражения рекомендуется составлять из величин одинакового типа, т.е. либо целых, либо вещественных. Показатели степени могут быть целыми константами как для целых, так и для вещественных оснований.

2. Вычисление арифметических выражений производится с учетом старшинства операций: сначала выполняются операции возведения в степень, затем операции умножения и деления и, наконец, все операции сложения и вычитания. Если в выражении встречаются подряд несколько умножений и делений, то действия выполняются подряд слева направо. Аналогично выполняются операции сложения и вычитания.

Например, выражение *A/B*C* означает $\frac{A \cdot C}{B}$, но не $\frac{A}{B \cdot C}$; а выражение *X-Y+Z* означает *(X-Y)+Z*, но не *X-(Y+Z)*.

3. Если в выражении встречаются подряд две последовательные операции возведения в степень, то вычисления выполняются справа налево.

Например, выражение *A**B**C* вычисляется в следующем порядке:

*B**C

*A**(B**C)*

Если показатель степени - небольшое целое число, то рекомендуется вместо функции возведения в степень использовать операцию умножения. Например:

$4\frac{R^3}{3}$ — *(4.*R*R*R)/3.*

По тем же причинам операция *X**5* будет более экономична, чем *X**5.*, так как в первом случае транслятор сводит эту операцию к последовательности умножений, а во втором случае использует библиотечные подпрограммы *EXP* и *ALOG*
*EXP(5.*ALOG(X))*.

4. Выражения, заключенные в круглые скобки, вычисляются в первую очередь. Если выражение, содержащее скобки, само заключено в круглые скобки, то вычисления производятся, начиная с внутренних, наиболее "глубоких" скобок.

Например, в выражении $A*(X-B/(X+C))$ сначала находится сумма $X+C$, затем частное $Y=\frac{B}{X+C}$, затем разность $Z=X-Y$, и наконец, произведение $A \cdot Z$.

Необходимо всегда следить за тем, чтобы число открытых скобок равнялось числу закрытых.

5. Нельзя ставить рядом два знака арифметических операций, а также опускать знак умножения между сомножителями.

Например, выражения $4AB$ и $\frac{X}{-Z}$ на ФОРТРАНЕ следует записывать так

$4.*A*B$ и $X/(-Z)$.

6. При вычислении выражений, состоящих из величин целого типа, надо помнить, что арифметические операции производятся с отбрасыванием дробных частей в промежуточных результатах.

Например, вычисляя выражение $4*(7/2)$, получим 12 (а не 14), так как частное от деления 7 на 2 будет равно 3, а не 3,5 как в обычной арифметике.

Рассмотрим несколько примеров.

Записать на ФОРТРАНЕ следующие арифметические выражения:

$(A+B)^4 \longrightarrow (A+B)**4$

$\frac{X+2}{Y+4} \longrightarrow (X+2.)/(Y+4.)$

$\left[(A+2B)^2-\frac{A}{4}\right]^8 \longrightarrow ((A+2.*B)**2-A/4.)**8$

$4AX-\frac{34{,}6\cdot 10^6}{735A} \longrightarrow 4.*A*X-34.6E6/(735.*A)$

Упражнение I.3. Расшифровать следующие арифметические выражения:

1) $A+B/(C*D)$

2) $X+Y/C*D$

3) $((A+B)/C)**3.2$

Упражнение I.4. Найти описки в записи следующих арифметических выражений:

1) $\frac{X+7}{Y-6} \longrightarrow X+7./Y-6$

2) $\left(\frac{A+B+C}{2X}\right)^3$ ⟶ `(A+B+C)/(2X)**3`

3) $\left(\frac{X}{Y}\right)^{R-2}$ ⟶ `(X/Y)**(R-2)`

§6. СТРУКТУРА ПРОГРАММ

Любая программа, записанная на языке ФОРТРАН, должна включать следующие элементы

Наименование программы	`PROGRAM....`

Описание переменных и массивов	`INTEGER...`
	`REAL.....`
	`DIMENSION...`

Тело программы	Система операторов
	`P1`
	`P2`
	`:::`
	`PN`
	`STOP`
	`END`

М е т к и

Любой из операторов может быть снабжен меткой (номером), которая должна состоять только из цифр и занимать не более пяти позиций.

Например, I2, 346, I75Ø3.

О П Е Р А Т О Р Ы

Командные операторы - это некоторые указания (инструкции), написанные в виде символа, ключевого слова или группы слов, данные машине для соответствующей обработки информации. Операторы записываются в той последовательности, какой требует решаемая задача, а их совокупность образует собственно программу вычислений.

§I. ОПЕРАТОР ПРИСВАИВАНИЯ

Пусть P - наименование переменной,

Q - арифметическое выражение.

Тогда общий вид арифметического оператора присваивания

$P=Q$.

При выполнении этого оператора машина сначала вычисляет значение арифметического выражения Q, а затем присваивает это значение переменной P. При этом знак равенства играет роль символа присваивания. Данный оператор может использоваться для присваивания переменным конкретных числовых значений, а также значений других переменных и арифметических выражений.

Например:

```
A=1.
X=Y
N=N+2
```

Формула $z=\frac{a^5}{y}\cos^2 x$ будет записана как:

```
Z=A**5/Y*COS(X)**2 .
```

Упражнение 2.I. Записать операторы присваивания для следующих формул:

1) $y=\frac{1}{m\sqrt{a\cdot b}}\cdot\cos|\pi x|$; 3) $y=\frac{m^{3/4}}{y^2+1}$;

2) $y=\ln\left(\frac{1}{\sqrt{S-32,5}}\cdot t^2\right)$; 4) $y=e^{a3x}\cdot\sin x^2$.

С помощью оператора присваивания можно преобразовывать целые величины в вещественные и наоборот. Варианты преобразований показаны в табл. 2.

Таблица 2

Тип переменной P	Арифметическое выражение Q		
	Целый	Вещественный	Удвоенной точности
Целый	Присвоение	Преобразование в целый	Преобразование в целый
Вещественный	Преобразование в вещественный	Присваивание	Преобразование в вещественный
Удвоенной точности	Преобразование в удвоен.точность	Преобразование в удвоен.точность	Присвоение

Например: для $I=2.5*A$, если $A=-3$, то $I=-7$.

Упражнение 2.2. Определить значения переменных A и K, которые получатся после выполнения операторов присваивания

1) $A=2/7$ 3) $K=19/4+5/4$

2) $A=19/4+5/4$ 4) $A=19.14.+5.14.$

§2. ПРОСТЕЙШИЕ ПРОГРАММЫ

При составлении программы на ФОРТРАНЕ в начале ее должен быть записан оператор *PROGRAM* вместе с наименованием этой программы. Наименование программы должно начинаться с буквы и состоять не более чем из восьми буквенных и цифровых символов, например, *LINT2*, *GERON*, *RE5* и т.д. В конце программы ставится оператор окончания *END*.

Рассмотрим следующий пример.

Составить программу для вычисления одного значения функции

$$y=\frac{(ax^5+b)^2-\ln(ax^5+b)}{\sqrt[3]{ax^5+b}}$$

для заданных вещественных величин a, b, x.

```
PROGRAM  POLIS
A=3.4
B=.725
```

```
X=6.
X=A*X**5+B
Y=X*X
Y=(Y-ALOG(X))/X**(1./3.)
STOP
END
```

Упражнение 2.3. Составить программу вычисления функции S для одного значения аргумента x = I,25, если

$$p=0{,}31e^{x}+x^{5};\quad q=\sin^{2}x-x;\quad S=\sqrt{p}+\ln|q-1{,}2\cdot10^{-3}|.$$

Рассмотрим еще пример.

Составить программу вычисления выражения

$$y=3{,}5\cdot\frac{IND}{1+IND^{2}}\cdot\sqrt{38{,}5+IND^{2}}$$

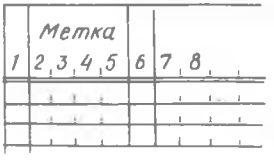
для одного значения $IND=0{,}72\cdot10^{2}$, заданного в явной форме как вещественная величина

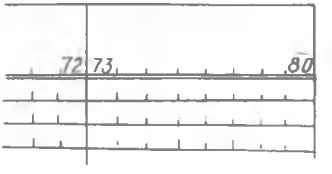
```
PROGRAM NEWS1
REAL IND, X, Y
IND=.72E+2
X=IND*IND
Y=3.5*IND*SQRT(38.5+X)/(1.+X)
STOP
END
```

§3. ЗАПИСЬ ПРОГРАММЫ НА БЛАНКЕ

Каждая программа, написанная на языке ФОРТРАН, состоит из отдельных операторов, сопровождающихся в некоторых случаях метками.

Программа обычно пишется на специальном бланке, каждая строка которого содержит 80 позиций.





При записи программы на бланке надо соблюдать следующие правила.

I. В каждой позиции записывается не более одного символа (буква, цифра, точка, запятая, скобка и др.), причем, некоторые позиции можно оставлять незаполненными.

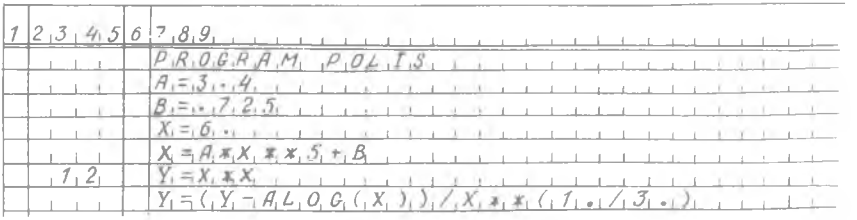
2. Каждый оператор обычно записывают на отдельной строке. Для записи операторов разрешается использовать только позиции с 7 по 72.

3. Если необходимо осуществить перенос записи операторов на следующую строку, то в шестой позиции этой строки надо поставить любой символ ФОРТРАНА, отличный от нуля и пробела.

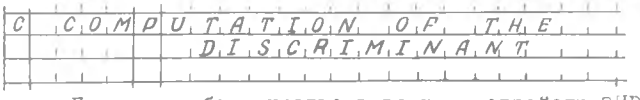
4. Метка операторов записывается только в I-5 позициях.

5. Один оператор(с продолжениями) может занимать не более 20 строк.

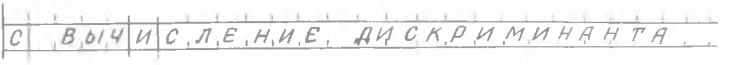
Рассмотрим запись на бланке программы одного из предыдущих примеров.



6. На бланке в любом месте можно писать комментарии, тогда в первой колонке этой строки ставится буква *C* .



Поскольку большинство выводных устройств ЭЦВМ имеют кроме латинского также и русский шрифт, это позволяет писать комментарии на русском языке.



Бланки ФОРТРАНА предназначены, главным образом, для удобства перфорирования программы. Одна строка бланка соответствует одной

80 - колонной перфокарте, при этом каждая позиция бланка соответствует одной колонке перфокарты.

§4. ОПЕРАТОРЫ УПРАВЛЕНИЯ

Операторы управления имеют несколько разновидностей. Их использование позволяет строить сложные программы с разветвлениями и соответствующими критериальными передачами управления.

Оператор перехода *GO TO*.

Этот оператор может употребляться в двух модификациях:

а) безусловной,

б) вычисляемой.

Оператор безусловного перехода имеет вид *GO TO* α, где α - метка (числовой набор от I до 5 позиций). При исполнении этого оператора происходит передача управления (переход) к оператору с меткой α .

Рассмотрим пример.

```
PROGRAM SUM
S=ϕ.
A=1.
12 S=S+1./(A*A)
 A=A+1.
GO TO 12
STOP
END
```

При исполнении этой программы будет вычисляться бесконечный ряд:

$S = 1/1^2 + 1/2^2 + 1/3^2 + \ldots$

Вычисляемый оператор перехода имеет вид:

GO TO ($\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$), I

Здесь в скобках помещены метки α_i операторов, а I - целочисленный индекс меток (I = 1, 2, 3, ... n). Количество меток обычно не ограничивается. Более того, одна и та же метка может появляться в списке несколько раз. Например, допустим оператор

GO TO (4,6,8,4,5,6), I

Вычисляемый оператор *GO TO* работает как переключатель, передавая управление на ту метку (и соответственно оператор), порядковая позиция которой указывается в данный момент индексом I. Например, если в приведенном выше операторе $I = 2$, то управление передается на метку 6 и т.д.

Рассмотрим следующий пример.

Предполагая, что K - целое число в интервале $0 \leqslant K \leqslant 29$, произвести вычисления:

если $0 \leqslant K \leqslant 9$, то $y = a \cdot e^{b+1}$;

если $10 \leqslant K \leqslant 19$, то $y = a \cdot b^2 + 2$;

если $20 \leqslant K \leqslant 29$, то $y = (a \cdot \sin b)^5$;

```
      PROGRAM SWITCH
      . . . . . .
      K=(0÷29)
      I = K/10+1
      GO TO (10,12,14), I
   10 Y = A * EXP(B+1.)
      GO TO 16
   12 Y = A * B * B + 2.
      GO TO 16
   14 Y = (A * SIN(B)) ** 5
   16 продолжение программы
```

Упражнение 2.4. Предполагая, что x - вещественное число в интервале $0,5 \leqslant x < 3,5$, произвести вычисления:

если $0,5 \leqslant x < 1,5$, то $y = (\sin x) \cdot (t \cdot \ln x)$;

если $1,5 \leqslant x < 2,5$, то $y = \ln x + x$;

если $2,5 \leqslant x < 3,5$, то $y = \sqrt{x}$.

Оператор условного перехода *IF* (арифметический)

Оператор условного перехода (передачи управления) имеет вид:

$IF(Q)\alpha_1, \alpha_2, \alpha_3$,

где Q - некоторое арифметическое выражение;

$\alpha_1, \alpha_2, \alpha_3$ - метки операторов.

Этот оператор вычисляет значение Q и затем передает управление по следующему правилу:

если $Q < 0$, то - оператору с меткой α_1;

если $Q = 0$, то - оператору с меткой α_2;

если $Q > 0$, то - оператору с меткой α_3.

С помощью этого оператора можно строить самые разнообразные программы.

Разветвляющиеся программы

При решении многих задач ход вычислительного процесса зависит от значения переменных, входящих в расчетные формулы. Иначе говоря, в зависимости от конкретной величины переменных задачи, на определенном этапе расчетов дальнейшие вычисления могут производиться по одному из двух или нескольких направлений. Такие вычислительные процессы называют разветвляющимися, и для них составляются разветвляющиеся программы.

Пример I. Составить программу определения x при условиях:

если $y < 1$, то $x = макс(a, в)$,

если $y \geq 1$, то $x = 1$

Блок-схема решения задачи приведена на рис. I.

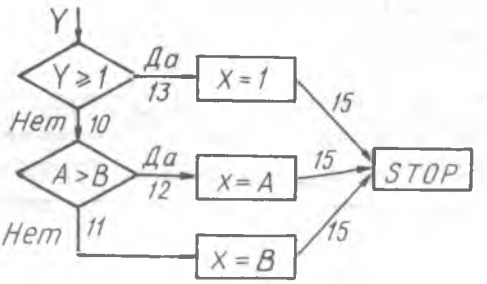


Рис. I.

```
      PROGRAM RASTR
      : : : : : : : : : : :
      IF(Y-1.)10,13,13
10    IF(A-B)11,11,12
11    X=B
      GO TO 15
12    X=A
      GO TO 15
13    X=1.
15    STOP
      END
```

Пример 2. Составить программу вычисления величины x,

$x=f(y)-\sqrt{z}$,

где $y=z+2$, при

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

Блок-схема решения задачи приведена на рис. 2.

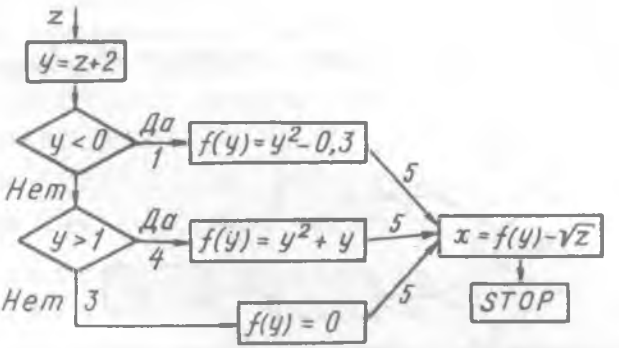


Рис. 2.

```
      PROGRAM STAR
      . . . . . .
      Y=Z+2.
      IF(Y)1,2,2
I     F=Y*Y-0.3
      GO TO 5
2     IF(Y-1.)3,3,4
3     F=0.0
      GO TO 5
4     F=Y*Y+Y
5     X=F-SQRT(Z)
      STOP
      END
```

Упражнение 2.5. Составить разветвляющуюся программу вычисления функции:

$$f(x,y)=\begin{cases}\sqrt{y-\dfrac{x}{b}}\,, & \text{если} \quad x+y\geq b\,;\\[2ex] \left(x-\dfrac{y}{b}\right)^2, & \text{если} \quad x+y<b\,.\end{cases}$$

Циклические программы

Большинство численных методов решения задач сводится к многократным повторениям некоторой группы операций, причем, обычно задается или число повторений циклов или условие их окончания. Такие вычислительные процессы называются <u>циклическими,</u> а многократно повторяющиеся участки программ <u>циклами.</u>

В зависимости от способа организации и назначения циклических процессов их подразделяют на несколько разновидностей, к числу которых относятся <u>простые циклы,</u> <u>итерационные циклы,</u> <u>кратные (вложенные) циклы.</u>

Простой цикл

В тех случаях, когда требуется повторить определенную группу операторов заданное число раз, можно использовать различные схемы счетчиков циклов; некоторые из них и рассматриваются ниже.

Пример 1. Составить программу вычисления факториала числа N

$$N! = N\cdot(N-1)\cdot(N-2)\cdot\ldots\cdot 2\cdot 1$$

I Вариант с прямым счетчиком

```
      IFACT=1
      K=1
1     IFACT=IFACT*K
      K=K+1
      IF(K-N)1,1,2
```

Прямой счетчик

2 Продолжение программы

II Вариант с обратным счетчиком

```
      INTEGER FACT
      FACT=1
      K=N
1     FACT=FACT*K
      K=K-1
      IF(K)2,2,1
```

Обратный счетчик

2 Продолжение программы

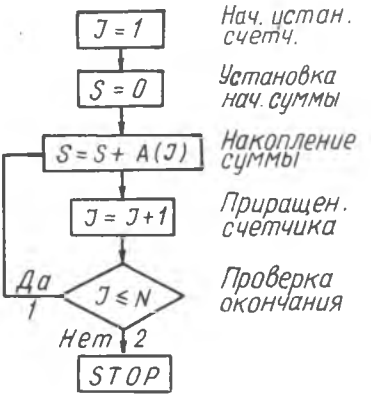
Рассмотрим пример программы на операции с массивами.

Пример 2. Составить программу вычисления суммы элементов массива A

$$S=\sum_{j=1}^{n} A_j.$$

Блок-схема решения задачи приведена на рис. 3.

```
      PROGRAM ART
      DIMENSION A(N)
      . . . . . .
      J=1
      S=ϕ.
1     S=S+A(J)
      J=J+1
2     IF(J-N)1,1,2
      STOP
      END
```



Р и с. 3.

И т е р а ц и о н н ы й ц и к л

Итерационные процессы характеризуются тем, что искомая величина y находится путем последовательных приближений. При этом каждое последующее значение y_{i+1} по сравнению с предыдущим y_i ближе к истинному значению y (для сходящихся процессов). Вычисления ведутся до получения заданной степени точности, т.е. до тех пор, пока не будет выполнено условие:

$$|y_{i+1} - y_i| \leqslant \varepsilon,$$

где ε - заданная степень точности.

Выделение модуля производится здесь в связи с тем, что получаемые разности $y_{i+1} - y_i$ могут оказаться отрицательными или знакопеременными.

Таким образом, в итерационных процессах количество повторений циклов заранее неизвестно. Поэтому вместо счетчиков циклов здесь нужно в конце каждого цикла предусматривать проверку окончания по вышеприведенному условию.

В некоторых случаях вычисления целесообразно производить не с абсолютной, а с относительной степенью точности, тогда условие окончания вычислений будет иметь вид:

$$\left|\frac{y_{i+1} - y_i}{y_{i+1}}\right| \leqslant \varepsilon.$$

П р и м е р I. Составить программу вычисления квадратного корня $y=\sqrt{a}$ по рекуррентной формуле

$$y_{i+1}=\frac{1}{2}\left(\frac{a}{y_i}+y_i\right)$$

с заданной степенью точности $|y_{i+1}-y_i|\leqslant\varepsilon$.

Обозначим: $Y1=y_i$ и $Y2=y_{i+1}$, за начальное (грубое) приближение примем значение y_0.

```
      PROGRAM KRONA
      A=<Значение a>
      E=<Значение ε>
      Y1=<Приближение y0>
    1 Y2=0.5(A/Y1+Y1)
      DELTA=ABS(Y2-Y1)
      Y1=Y2
      IF(E-DELTA)1,2,2
    2 STOP
      END
```

К итерационным циклам относится также вычисление сходящихся рядов с заданной степенью точности. В этом случае каждый вычисленный член ряда сравнивается со значением ε, и накопление суммы членов ряда производится до выполнения условия $|n|\leqslant\varepsilon$ (или лучше $|(n)/S_n|\leqslant\varepsilon$), где n -величина очередного члена ряда, а S_n - накопленная сумма.

Примечание: если все члены ряда положительны, то модуль их выделять необязательно.

П р и м е р 2. Вычислить сумму членов ряда S в области значений ($0<x\leqslant 1$)

$$S=\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}+\ldots$$

с заданной относительной степенью точности

$$\left(\frac{x^n}{n!}\right)/S_n\leqslant\varepsilon,$$

где S_n - накапливаемая сумма членов ряда.

Исходя из отношений двух соседних членов ряда

$$a_n:a_{n-1}=\frac{x^n}{n!}:\frac{x^{n-1}}{(n-1)!}=\frac{x}{n},$$

получим рекуррентное отношение

$a_n = a_{n-1} \cdot \frac{x}{n}$,

которое удобно использовать при составлении операторной программы.

```
    PROGRAM ITERAL
    X=<Значение x>
    E=<Значение ε>
    R=1.
    B=1.
    S=ϕ.
5   R=R*X/B
    S=S+R
    B=B+1.
    IF(E-R/S)5,7,7
7   STOP
    END
```

Для случая знакопеременного ряда в рассмотренную программу нужно ввести следующую конструкцию

```
. . . . . .
R=-1.
. . . . . .
. . . . . .
R=ϕ.-R*X/B
```

Таким образом, в общем случае при необходимости организации знакопеременного вычислительного процесса можно вводить дополнительные операторы:

```
D=1. (или -1.)
. . . . .
D=-D
```

Здесь получаемое в каждом цикле значение D будет знакопеременной единицей.

Упражнение 2.6. Составить программу табулирования функции

$F(x) = \ln \frac{1}{x} (\sqrt{ax} + b)$

для значений a = 2,375, b = 0,56 и аргумента, изменяющегося в интервале $0{,}5 \leqslant x \leqslant 2{,}5$ с шагом h = 0,1.

О п е р а т о р ц и к л а *DO*

Этот оператор позволяет строить простые циклические программы. С помощью оператора *DO* можно выполнить заданное число раз

некоторую группу операторов, следующих непосредственно за ним. Общий вид оператора цикла:

$DO\ \alpha\ N = n_1, n_2, n_3$,

где α – метка последнего оператора повторяемой группы (обычно это оператор *CONTINUE*);

N – параметр цикла (простая целая переменная) является счетчиком цикла;

n_1, n_2, n_3 – натуральные ненулевые числа ($n_1 \leqslant n_2$) или целые переменные, которым к моменту выполнения оператора *DO* были присвоены значения некоторых натуральных чисел.

Оператор *DO*, оператор *CONTINUE* с меткой α и все операторы между ними составляют цикл. Операторы данного цикла выполняются сначала при $N = n_1$, затем – при $N = n_1 + n_3$, далее – при $N = n_1 + 2n_3$ и т.д. до тех пор, пока $N \leqslant n_2$. Таким образом,

n_1 – начальное значение параметра цикла,

n_2 – конечное значение параметра цикла,

n_3 – приращение цикла.

Если приращение цикла $n_3 = 1$, то оператор *DO* можно представить в сокращенной записи:

$DO\ \alpha\ N = n_1, n_2$.

Например:

```
      DO 4 N=1,25
      :
   <25 раз>
      :
4  CONTINUE
```

```
      DO 6 K=2,5ϕ,2
      :
   <25 раз>
      :
6  CONTINUE
```

Оператор *CONTINUE* является "пустым" оператором. Он не указывает на выполнение каких-либо команд, но он может иметь метку и таким образом указывать место конца цикла. Роль оператора *CONTINUE* может выполнять также обыкновенный арифметический оператор присваивания.

Количество повторений цикла в общем случае может быть определено по формуле:

$$n = \frac{x_n - x_1}{\Delta x} + 1,$$

где x_n, x_1 - верхняя и нижняя границы изменения переменной x,

Δx - приращение переменной.

Так как величины n_1, n_2, n_3 могут быть также целыми переменными, допустимы следующие операторы цикла:

```
DO 15 IND=K,52
DO 24 J=M,LIM.
```

Обобщая сказанное, можно сформулировать основные правила записи операторов цикла:

1) величины n_1, n_2, n_3 могут быть только целыми константами без знака или целыми простыми переменными;

2) параметр цикла N может принимать только положительные значения (в частности, цикл нельзя начинать с нуля);

3) шаг цикла не может быть отрицательным.

Примеры неправильной записи оператора *DO*:

```
DO 16 I=ϕ,5ϕ
```

- цикл не может начинаться с нуля;

```
DO 23 MATR=25,5,-1
```

- шаг цикла не может быть отрицательным, и, кроме того, n_2 не должно быть меньше n_1;

```
DO 34 K=2,A,3
```

- параметры цикла должны быть целыми величинами;

```
DO 27 J=N,MAX-1
```

- выражения в записи оператора цикла не допускаются.

После выполнения последнего оператора цикла *DO* машина должна перейти к выполнению первого оператора цикла либо выйти из цикла, в зависимости от значения счетчика.

Рассмотрим примеры построения циклических программ.

П р о с т о й ц и к л

П р и м е р I. Составить программу вычисления ряда:

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{10}$$

Фрагмент программы будет иметь вид:

```
      S=0.
      DO 2 N=1,10
      R=N                    (Преобразование типа переменной)
      S=S+1./R
    2 CONTINUE
```

Пример 2. Составить программу табулирования функции $F(x)=x+\sin x$ для n значений аргумента, изменяющегося в интервале $0 \leqslant x \leqslant 1$ с шагом h = 0,1.

Рассчитываем число циклов $n=11$.

Программа будет иметь следующий вид:

```
      PROGRAM LIST
      DIMENSION F(11)
      H=0.1
      X=0.
      DO 1 I=1,11
      F(I)=X+SIN(X)
      X=X+H
    1 CONTINUE
      STOP
      END
```

Циклы с массивами

Пример 3. Одномерный массив A состоит из 200 элементов. Возвести в квадрат каждый нечетный элемент массива и в третью степень - каждый четный.

```
DIMENSION A(200)
DO 3 J=2,200,2
A(J-1)=A(J-1)**2
A(J) =A(J)** 3
3 CONTINUE.
```

Упражнение 2.7. Составить программу нахождения минимального по абсолютной величине значения $X(N)$ в массиве из 500 элементов. Найденное значение присвоить величине AMN. Первое число в массиве считать положительным. Элементы массива сохранить.

В некоторых программах треоуется использование обратного счетчика циклов, а в цикле *DO* такая возможность не предусмотрена. В этих случаях приходится строить вспомогательные конструкции.

П р и м е р 4. Составить программу переписи массива *M1*(20) в обратном порядке.

```
DIMENSION M1(2ф), M2(2ф)
DO 2 I=1,2ф
K=21-I (Обратный счетчик)
M2(K)=M1(I)
2 CONTINUE
```

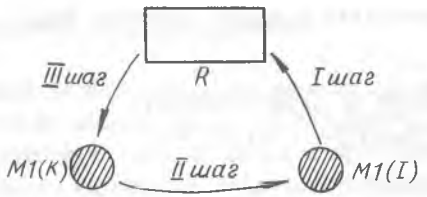
В данной программе инвертированный массив будет уже иметь другое наименование - *M2*, и в памяти машины образуются два массива: один - с прямым расположением элементов *(M1)*, другой - с обратным *(M2)*.

Если же нужно произвести перемещение элементов внутри одного и того же массива *M1*, то можно дополнить приведенную программу еще одним циклом переписи массива *M2* (2ф) в *M1* (2ф). Однако более интересным представляется следующий вариант программы:

```
DIMENSION M1(2ф)
DO 2 I=1,1ф
K=21-I
R=M1(I)
M1(I)=M1(K)
M1(K)=R
2 CONTINUE
```

} Пересылка

Здесь применена схема перемещения элементов массива с использованием так называемого "посредника" *R*



В качестве "посредника" берется свободная ячейка с именем R, которая служит как бы промежуточной памятью для сохранения перемещаемых элементов массива. При этом число циклов должно быть вдвое меньше количества элементов массива.

П р и м е р 5. Составить программу вычисления непрерывной (цепной) дроби при заданном x

$$f=\cfrac{x}{1-\cfrac{x^2}{3-\cfrac{x^2}{5-\cfrac{x^2}{7-\cfrac{x^2}{9-\cfrac{x^2}{11}}}}}}$$

Представим рекуррентное соотношение дроби, как

$$z_i = a_i - \frac{x^2}{z_{i-1}} .$$

Тогда программа будет иметь вид:

```
PROGRAM RITM
X=2.5
Z=11.
DO 3 I=1,5
A=11-2*I (Обратный счетчик)
Z=A-X*X/Z (Вычисление z_i )
3 CONTINUE
F=X/Z
STOP
END
```

В ы ч и с л е н и е п о л и н о м о в

Если требуется вычислить значение полинома вида:

$$P=\sum_{K=0}^{n} a_K x^K = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 ,$$

то его обычно преобразуют по схеме Горнера:

$$P=(\ldots((\underbrace{(a_n x + a_{n-1})}_{1\text{цикл}} x + a_{n-2}) x + \ldots + a_1) x + a_0 .$$

2 цикл

Пусть n = 5, тогда

$$P=((((a_5 x + a_4) x + a_3) x + a_2) x + a_1) x + a_0 .$$

Таким образом, получим: число циклов n = 5, количество коэффициентов K = 6.

Так как в цикле DO нулевые индексы употреблять нельзя, сдвигаем индексы при коэффициентах a_i на один порядок и принимаем $P_{нач} = a_6$.

```
      PROGRAM POL
      DIMENSION A(6)
      . . . . . .
      P=A(6)
      DO 3 I=1,5
      K=6-I
      P=P*X+A(K)
    3 CONTINUE
      STOP
      END
```

Частным случаем рассмотренной задачи является вычисление полинома вида:

$$P=\sum_{i=1}^{n} x^i .$$

Его можно привести к схеме:

$$P=(\ldots(((P_0+1)x+1)x+1)x+\ldots,$$

где $P_0 = 0$. Тогда получим программу:

```
      . . . . . .
      P=0.
      DO 12 I=1,N
      P=(P+1)*X
   12 CONTINUE
      STOP
      END
```

Построение экономичных циклов

При программировании циклических процессов следует обращать внимание на их экономичность. В частности, нужно избегать громоздких вычислений внутри циклов и при возможности выносить их за пределы цикла. Рассмотрим следующий фрагмент циклической программы:

```
      DO 10 N=1,1000
      X(N)=2.0*(G+ALPHA)+Y(N)
   10 CONTINUE
```

в приведенном примере выражение *2.ϕ*(G+ALPHA)* будет вычисляться каждый раз при выполнении цикла. Целесообразно представить эту программу следующим образом:

```
BETA=2.ϕ*(G+ALPHA)
DO 1ϕ N=1,1ϕϕ
X(N)=BETA+Y(N)
1ϕ CONTINUE
```

Вложенные циклы

Цикл *DO* может содержать внутри один или несколько других циклов *DO*. При этом не допускается "перекрытий" циклов: включаемый цикл - вложенный - должен полностью находиться внутри включающего внешнего. Приведем пример допустимого включения двух циклов:

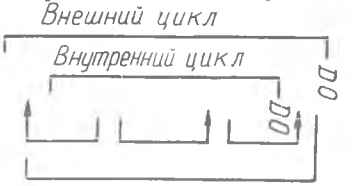
```
S=ϕ.
DO 1 I=1,12
DO 2 K=1,5
S=S+AR(I,K)
2 CONTINUE
1 CONTINUE
```

Эта программа вычисляет сумму элементов матрицы *AR* размерностью 12x5.

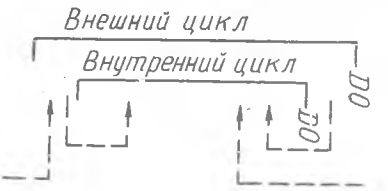
Приведенную программу можно упростить, если оба цикла будут иметь общее окончание:

```
S=ϕ.
DO 1 I=1,12
DO 1 K=1,5
S=S+AR(I,K)
1 CONTINUE
```

На рис. 4 схематически изображены допустимые, а на рис. 5 - недопустимые передачи управления для вложенных циклов.



Р и с. 4.



Р и с. 5.

Из схем видно, что вход внутрь цикла возможен только через его начало, а выход - из любого места.

Рассмотрим еще один пример с использованием вложенного цикла для случая задачи сортировки.

Составить программу расстановки элементов массива $T(N)$ в порядке возрастания для N = 200 элементов массива.

Алгоритм программы таков, что в каждом внутреннем цикле сравниваются два соседних элемента массива и, если необходимый порядок (возрастание) нарушен, пара переставляется местами. Таким образом, при первом прохождении внутреннего цикла $T(1)$ сравнивается (и переставляется в случае необходимости) с $T(2)$, при втором - $T(2)$ с $T(3)$, далее $T(3)$ с $T(4)$ и т.д.

После N повторений внутреннего цикла наибольшее число оказывается в конце массива. Этим заканчивается первое прохождение внешнего цикла. Теперь необходимо повторить внутренний цикл $N-1$ раз, так как последний элемент массива не учитывается и т.д. В такой программе текущее значение индекса внешнего цикла будет служить верхней границей индекса внутреннего цикла.

I Вариант

```
    N=200
    DO 2 I=1,N
    K=N-I
    DO 2 J=1,K
    IF(T(J)-T(J+1))2,2,1
1   R=T(J)
    T(J)=T(J+1)
    T(J+1)=R
2   CONTINUE
```

2 Вариант

```
    N=200
    DO 2 I=1,199
    K=1+I
    DO 2 J=K,N
    IF(T(I)-T(J))2,2,1
1   R=T(I)
    T(I)=T(J)
    T(J)=R
2   CONTINUE
```

Разберите самостоятельно II-й вариант программы и определите, какой в данном случае использован алгоритм.

§5. СЛУЖЕБНЫЕ ОПЕРАТОРЫ

О п е р а т о р *STOP*

При исполнении этого оператора машина останавливается так, что работа по программе может быть возобновлена лишь с самого начала. Это так называемый полный останов. Одновременно с остановом машины печатается константа оператора *STOP* , если она указана (например, *STOP* 1001).

О п е р а т о р *PAUSE*

При исполнении оператора *PAUSE* машина тоже останавливается, печатая слово *PAUSE* и константу, если она указана.

Оператор *PAUSE* отличается от *STOP* тем, что после *PAUSE* работа программы может быть продолжена нажатием кнопки "ПУСК" на пульте машины.

Оператор *PAUSE* может быть использован как эффективное средство отладки или контроля программ с пульта, а также при смене лент, дисков и т.п.

§6. ОПЕРАТОРЫ ВВОДА-ВЫВОДА ИНФОРМАЦИИ

Для ввода исходных числовых значений величин можно в начале программы использовать группу операторов присваивания.

Например:

A = 5.316
B = .φ125
N = 1

Операторы присваивания обычно применяются в тех случаях, когда числовые данные являются постоянными для рассматриваемой программы или когда программа используется только один раз при небольшом количестве цифровых данных.

В тех случаях, когда нужно вводить большое количество чисел или когда программа используется многократно, числовые данные обычно вводятся с перфокарт при помощи специального оператора вво-

да:

READ(β,α)<Список>

Аналогично, для вывода информации используется оператор вывода:

WRITE(β,α)<Список>

В этих операторах:

β - либо целое положительное число, либо переменная целого типа ($1 \leqslant \beta \leqslant 15$), эта величина представляет собой номер какого-либо устройства ввода-вывода.

Например: *β* = 1 - ввод с перфокарт,

β = 2 - вывод на перфокарты ,

β = 3 - вывод на АЦПУ.

α - метка специального оператора *FORMAT*, определяющего форму ввода-вывода переменных,

"Список" - идентификаторы переменных и массивов, разделенных запятыми.

Например:

READ(1,25)IMP, A(5,4), F42

WRITE(3,12)X, Y, SUM

При вводе или выводе массива величин достаточно в операторах *READ* или *WRITE* указать его наименование.

Например:

DIMENSION X(25)

. . . . . .

READ (1,6)X

Если необходимо произвести ввод или вывод элементов массива в последовательности, отличной от принятой в ФОРТРАНЕ, то можно воспользоваться методом неявного цикла.

Например, оператором *READ(1,6)(X(I),I=3,12)* будет вводиться массив *X(3), X(4), ..., X(12)*, так как здесь указан интервал изменения индекса $3 \leqslant I \leqslant 12$.

Двухмерный массив можно ввести, например, оператором

READ(1,7)((A(I,J), I=1,1ϕ), J=1,2ϕ)

Этот оператор введет элементы массива в следующем порядке:

A(1,1), A(2,1), ... , A(1ϕ,1)

A(1,2), A(2,2), ... , A(1ϕ,2)

. . . . . .

A(1,2ϕ), A(2,2ϕ), ... , A(1ϕ,2ϕ)

По существу, здесь оператор *READ* для каждого значения внешнего индекса *J* повторяет обычную индексацию по внутреннему индексу I.

Конструкция *READ(1,8)(RA(I,J),J=1,4)* - указывает первые четыре элемента I-й строки матрицы *RA* :

RA(1,1) RA(1,2) RA(1,3) RA(1,4)

Аналогично записываются операторы вывода информации. Например, оператор *WRITE(3,15)I,A,(X(L),L=1,10)* печатает 12 величин: целую I, вещественную *A* и 10 вещественных чисел массива *X(L)*.

Конструкция *WRITE(3,16)(RA(K,K),K=2,6)* - перечисляет диагональные элементы матрицы *RA* :

RA(2,2) RA(3,3) RA(4,4) RA(5,5) RA(6,6)

Использование неявного цикла позволяет также ввести или вывести не все элементы массива.

Например:

```
DIMENSION A(12)
. . . . . .
READ(1,1)(A(I), I=2,5)
```

Здесь будут введены элементы массива *A* только со второго по пятый.

П р и м е ч а н и е. Следует обратить внимание на то, что поскольку двухмерные матрицы размещаются в памяти машины по столбцам, то и размещение исходных данных на перфокартах также должно производиться по столбцам.

§7. ОПЕРАТОР *FORMAT*

Неисполняемый оператор *FORMAT* снабжает операторы ввода и вывода информацией о форме вводимых данных и выводимых результатов. Каждый оператор *FORMAT* должен обеспечивать работу соответствующего оператора ввода или вывода. Поэтому операторы *FORMAT* всегда имеют метку - α .

С помощью этого оператора можно указать форму представления чисел, количество значащих цифр числа и знаков после запятой. Общий вид оператора:

$$\alpha\ FORMAT\ (\underbrace{C_1, C_2, \ldots, C_n}_{Форматы})$$

Практически используются следующие форматы:

nIw - представление целых чисел;

nFw.d } - представление вещественных чисел;
nEw.d }

nDw.d - представление чисел с удвоенной точностью;

wX - представление пробелов;

'ABC...' - представление текста (литерал);

Tw - позиционный код;

A - символьный код;

Здесь *n* - коэффициент кратности;

w - число позиций в поле формата;

d - число цифр справа от десятичной точки.

Рассмотрим особенности каждого формата.

I. Формат I для ввода-вывода целых чисел. При использовании этого формата число позиций *w* должно быть не менее количества значащих цифр числа.

Например, операторы
```
READ(1,5)K
5 FORMAT(I8)
```
обеспечивают чтение с перфокарт восьмизначного числа *K* .

Если формат задан неправильно, т.е. число при выводе не умещается в *w* позиций, то все поле забивается звездочками **...

П р и м е р:
```
WRITE(3,2)N
2 FORMAT(I5),
```
если число *N* = + 29, то ␣␣␣29;
если число *N* = + 29375, то 29375;
если число *N* = - 29, то ␣␣- 29;
если число *N* = - 29375, то *****

2. Формат *F* для ввода-вывода вещественных чисел в форме основной константы, (фиксированная запятая).

При использовании формата *Fw.d* из общего числа позиций *w* будет отведено *d* позиций для дробной части.

Например, оператор
```
DIMENSION A(5)
READ (1,3)A
3 FORMAT(5F6.2)
```

обеспечивает чтение с перфокарты пяти шестизначных чисел массива A с запятой, отделяющей в них двухзначную дробную часть.

При вводе чисел десятичная точка может не перфорироваться на карте. Если она есть, то количество позиций W должно включать и ее позицию. В случае несоответствия позиций десятичной точки на карте и указанной форматом, число вводится таким, каким оно отперфорировано.

Рассмотрим пример формата вывода чисел с фиксированной запятой
```
WRITE(3,4)X
4 FORMAT(F8.3),
```

Если число X = 65,7 то ␣␣65.700;
если число X = - 37,74, то ␣-37.740;
если число X = 1974,11, то 1974.110;
если число X = - 1974,11, то ********

3. Формат E для ввода-вывода вещественных чисел в экспоненциальной форме.

При использовании формата $Ew.d$ из общего числа позиций w четыре отводятся для отображения порядка (например, $E+04$), поэтому должно соблюдаться условие $w>d+4$.

П р и м е р:
```
WRITE(3,7)S
7 FORMAT(E12.6) .
```

Если число S = 65,7897, то согласно формату 7 будет выведено: 0.657897 E + 02.

Так как кроме порядка нужно отвести позиции также под знак, точку и ноль целых, то лучше соблюдать условие $W>d+7$.

4. Формат D аналогичен формату E и служит для ввода-вывода данных удвоенной разрядности.

Например:
```
WRITE(3,7)R
7 FORMAT(D17,10)
```

5. Формат WX позволяет ввести или вывести поле, состоящее из W пробелов.

Например:
```
WRITE(3,6)S
6 FORMAT(6X,F6.2)
```

Если S = 238.5, то печатается ␣␣␣␣238.50. Число пробелов будет на один меньше, так как первый символ является управляющим.

6. Используя литерал $'ABC.'$, можно вывести на печать данные

в том виде, как они записаны в операторе *FORMAT* . При этом допустимы все символы АЦПУ.

Пример: *'X='*

'ЭТО ЗАГОЛОВОК'

7. Позиционный код *TW* указывает позицию в записи, с которой должна быть начата передача данных. При этом фактическая позиция начала текста будет на единицу меньше *W* из-за того, что первый символ является управляющим, т.е. *(W-1)* .

Например:

```
5 FORMAT('ФPART NO 1ФФ95', T4Ф,'1977 REPORT', T8Ф,
                'DECEMBER')

Поз.1              Поз.39              Поз.79
PART NO 1ФФ95      1977·REPORT         DECEMBER
```

Максимальная длина строки при печати может содержать не более 120 позиций.

Рассмотрим пример вывода нескольких величин с разными форматами :

```
      WRITE (3,17) N, V, S
17    FORMAT (I5, 2X, 'V=', E11.4, 3X, 'S=', F6.3)
```

Если значения (*N* = 25, *V* = 6,349, *S* = 0,157482), то при печати получим:

␣␣25␣␣V=␣Ф.6349E+Ф1␣␣␣S=␣Ф.157.

Следует отметить, что переход на новую строку печати (или на новую перфокарту) осуществляется в том случае, если в спецификации *FORMAT'а* встречается символ / , а также если использование спецификации доходит до последней закрывающейся скобки.

Рассмотрим случай ввода 20 элементов массива *A* , записанных на двух перфокартах по 10 величин формата *F6.2*

```
   READ (1,4) A
4  FORMAT (1ØF6.2)
```

Если же нужно, к примеру, ввести шесть чисел формата *F1Ø.5* с одной перфокарты и восемь чисел формата *F6.4* - с другой, то используется символ / перехода на следующую перфокарту:

```
   READ (1,2) A, B
2  FORMAT (6F1Ø.5/8F6.4)
```

П р и м е р. Вывести на печать столбцом числовые значения переменных

$N=1328;\ A=23,872;\ B=0,7\cdot10^{-4}$

Это можно осуществить, например, так:

```
   WRITE(3,5)N,A,B
5  FORMAT(1ФX,'N=',I6/1ФX,'A=',F6.3/1ФX,'B=',E8.2).
```

Такая запись *FORMAT'a* предписывает выдачу на печать заданных переменных в следующем виде:

```
␣.....␣N=  ␣␣ 1328
␣.....␣A= 23.872
␣.....␣B= Ф.7ФЕ-Ф4
```

Для получения двойного интервала между строками ставят две наклонные черты - // .

Г р у п п а Ф о р м а т о в

Чтобы повторить группу форматов, ее необходимо заключить в круглые скобки, а перед группой поставить указатель повторения. Если указатель отсутствует, то он полагается равным единице. С помощью группы повторения можно управлять порядком использования форматов, когда список ввода-вывода больше, чем форматов в операторе *FORMAT*.

Например, операторы:

```
   READ(1,15)N,A,B,C,D,E
15 FORMAT(I4,3(F6.2,D1Ф.3))
```

приводят к следующей операции чтения

$$\underbrace{N(I4)\,A(F6.2)\,B(D1\Phi.3)}_{1\ \text{запись}},\quad \underbrace{C(F6.2)\,D(D1\Phi.3)}_{2\ \text{запись}}\quad \underbrace{E(F6.2)}_{3\ \text{запись}}$$

Если в рассматриваемом примере в операторе *FORMAT* исключить указатель повторения, т.е.

```
15 FORMAT(I4,(F6.2,D1Ф.3)),
```

то чтение будет произведено с трех перфокарт

```
N(I4)A(F6.2)B(D1Ф.3)      - 1 запись
C(F6.2)D(D1Ф.3)           - 2 запись
E(F6.2)                   - 3 запись
```

Аналогично производится распечатка данных при выводе.

П р и м е р. Вывести на печать величину S и элементы матрицы X размерностью 3x4 построчно.

Операторы: `WRITE(3,5)S,((X(I,J),J=1,4),I=1,3)`

`5 FORMAT(7X,F4.2/4(F8.5))`

обеспечат следующий вид печати:

␣␣␣␣␣␣␣S

X(1,1)X(1,2)X(1,3)X(1,4)

X(2,1)X(2,2)X(2,3)X(2,4)

X(3,1)X(3,2)X(3,3)X(3,4)

В в о д - в ы в о д м а с с и в о в д а н н ы х

В приведенных ранее примерах были показаны некоторые приемы организации ввода-вывода массивов данных. Отметим, что наиболее часто используется достаточно простой способ ввода-вывода явно описанных массивов, например:

```
DIMENSION A(24)
. . . . . .
READ(1,7)A
7 FORMAT(8F8.5)
. . . . . .
```

В этом случае с трех перфокарт будет введен массив из 24 чисел формата $F8.5$. Эти числа будут присвоены индексированным переменным A (1), A (2), A (3) и т.д.

В отдельных случаях может быть использован описанный ранее метод неявного цикла и групповых форматов.

Если по каким-либо причинам массив данных не описывается оператором $DIMENSION$ и в программе элементы массива употребляются как неиндексированные переменные, можно рекомендовать способ циклического обращения к переменным массива:

I В а р и а н т

```
. . . . . .
   K=0
3  READ(1,5)X
5  FORMAT(F10.5)
. . . . . .
   K=K+1
   IF(K-N)3,7,7
. . . . . .
7  продолжение программы
```

II В а р и а н т

```
. . . . . .
   DO 5 I=1,N
   READ(1,3)X
3  FORMAT(F10.5)
. . . . . .
5  CONTINUE
   продолжение программы
```

Здесь в каждом цикле обращения к оператору *READ* будет вводиться очередное значение *X* из общего заданного массива чисел. При этом каждое число должно быть записано на отдельной перфокарте. Поэтому данный способ можно рекомендовать лишь для обработки сравнительно небольших массивов (порядка 5-20 чисел).

Аналогично, при печати вычисляемого массива величин операторы вывода *WRITE* и *FORMAT* могут помещаться внутрь цикла, и при каждом его повторении будет печататься очередное значение переменной.

8. Символьный код *A* используется для передачи данных, которые запоминаются в основной памяти в символьной форме. Количество символов, передаваемых по форматному коду *A*, зависит от длины соответствующей переменной в списке ввода-вывода, т.е. равно 4 для переменных целого и вещественного типа и 8 для удвоенной точности.

Например, для вывода строки переменных *A*, *B*, *C*, где *A*, *B* - вещественные, а *C* - удвоенной точности, записываются операторы:

```
    WRITE(3,26)A,B,C
26  FORMAT(3X,A6,A5,A6).
```

Если $\underbrace{A1B2}_{A}$, $\underbrace{C3D4}_{B}$ $\underbrace{E5F6G7H8}_{C}$,
то печатается:

␣␣A1B2|␣C3D4|E5F6G7.

§8. РЕДАКТИРОВАНИЕ РЕЗУЛЬТАТОВ

При исполнении оператора вывода информации машина не печатает первый знак строки. Этот знак (символ) является управляющим сигналом для печатающего устройства и определяет специфику печати следующим образом:

- ␣ - перевод на одну строку,
- 0 - перевод на две строки,
- 1 - перевод на следующую страницу, (1 страница = 63 строки),
- \+ - не продвигать бумагу.

Возможности редактирования результатов с помощью первого сим-

вола строки достаточно широки, но их использование требует внимательности при формировании оператора *FORMAT*.

П р и м е р. Операторы

```
    WRITE(3,15) A
15  FORMAT(4X,F10.2)
```

печатают на следующей строке десятиразрядную величину *A*, начиная с четвертой позиции строки. Переход к следующей строке обеспечивается благодаря тому, что первый символ строки является пробелом.

Перевод на следующую строку можно осуществить также следующим способом:

```
    WRITE(3,15)A
15  FORMAT('␣', F10.2)
```

Операторы:

```
    WRITE(3,5)
5   FORMAT('16 REDACTOR')
```

позволяют перейти на новую страницу и напечатать текст:

```
 6  REDACTOR
```

Рассмотрим основные принципы обработки информации операторами ввода-вывода на следующем примере.

Заданы операторы:

```
    WRITE(3,24)X
24  FORMAT('␣X=', F7.2,8X,'Y=', F6.4).
```

Порядок их исполнения будет следующим.

Производится анализ спецификации оператора *FORMAT* :

1) обрабатывается литерал *'␣X='* . Так как первым символом является пробел, то бумага продвигается на одну строку, и затем печатается текст *X=* ;

2) управление передается на формат *F7.2* . Происходит обращение к списку оператора *WRITE* , в котором находится элемент *X* . По идентификатору элемента из соответствующей ячейки памяти машины выбирается информация, преобразуется в соответствии с форматом *F* , и получившимися значениями заполняются позиции с третьей по девятую;

3) обрабатывается код *8X* . Позиции с 10-й по 17-ю остаются пустыми (пробелы);

4) обрабатывается литерал *'Y='* . В позициях 18-й и 19-й помещаются символы *Y=* ;

5) обрабатывается формат *F6.4* . Происходит обращение к списку оператора *WRITE* . Поскольку список исчерпан, оставшаяся часть строки будет пустой, и работа операторов заканчивается.

Таким образом, если X = - 23.17, получим:

$X=$␣-23.17␣...␣ $Y=$␣␣.....

8 пробелов

Если использование спецификации оператора *FORMAT* доходит до последней закрывающейся скобки, то обработка текущей единицы записи заканчивается и производится проверка: имеются ли еще необработанные элементы в списке оператора ввода-вывода. Если таких элементов нет, работа оператора заканчивается. Если список оператора ввода-вывода не исчерпан, то управление обменом информации начинает новую единицу записи и продолжает чтение спецификаций, начиная с внутренней открывающей скобки или, при отсутствии таковой, с открывающей внешней скобки.

Рассмотрим примеры решения задач с использованием операторов ввода-вывода информации.

П р и м е р I. Составить программу для вычисления значений функции $f(x)=e^x+3x-1$ в I2 точках x_i , если известно, что все x_i пробиты на одной перфокарте.

x_i = 0.I00; 0.I09; 0.205; ...; 0.375.

```
      PROGRAM TABLE
      DIMENSION X(12), F(12)
      READ(1,2)X
    2 FORMAT(12F6.3)
      DO 4 K=1,12
      F(K)=EXP(X(K))+3.*X(K)-1.
    4 CONTINUE
      WRITE(3,6)X,F
    6 FORMAT(6X,'X=',12F8.4/3X,'F(X)=',12F8.4)
      STOP
      END
```

Решение задачи будет отпечатано в следующем виде:

```
X = Ø.IØØØ   Ø.IØ9Ø  . . .  Ø.375Ø

F(X)= 2.4Ø52   2.4422  . . .  3.58ØØ
```

Если в этой же программе операторы вывода результатов записать в другой форме:

```
      WRITE(3,6)X(K),F(K)
6   FORMAT(6X,'X=',F8.4,3X,'F(X)=',F8.4)
```

и поместить их внутрь цикла (перед оператором *CONTINUE*), то результаты решения будут отпечатаны в два столбца:

X = Ø.IØØØ	F(X)= 2.4Ø52
X = Ø.IØ9Ø	F(X)= 2.4422
. . . . .	. . . . .
X = Ø.375Ø	F(X)= 3.58ØØ

П р и м е р 2. Составить программу вычисления и печати таблицы функции двух переменных

$$y=\frac{7x^2\cdot e^{-zx}}{2+z\cdot|x|}$$

для $X = -5, -4, \ldots, +5$; $z = 0.01; 0.02; 0.05; 0.10; 0.20; 0.35$

```
      PROGRAM TABUL
      DIMENSION Z(6)
      READ(1,1)(Z(I),I=1,6)
    1 FORMAT(6F6.2)
      X=-5.
    2 WRITE(3,3)X
    3 FORMAT(4X,'X=',F3.Ø)
      DO 5 I=1,5
      Y=7.*X**2*EXP(-Z(I)*X)/2.+Z(I)*ABS(X))
      WRITE(3,4)Y
    4 FORMAT(4X,'Y=',E12.6)
    5 CONTINUE
      X=X+1.
      IF(X-5.1)2,6,6
    6 STOP
      END
```

П р и м е р 3. Составить программу вычисления суммы элементов главной диагонали матрицы *M* размерностью 6x8 (элементы целые, двухразрядные)

```
      PROGRAM MATR
      DIMENSION M(6,8)
      READ(1,2)M
    2 FORMAT(24I3)
      N=0
      DO 3 I=1,6
      DO 3 J=1,8
      IF(I-J)3,4,3
    4 N=N+M(I,J)
    3 CONTINUE
      WRITE(3,5)N
    5 FORMAT(3X,'N=',I4)
      STOP
      END
```

Эту программу можно упростить, если исключить внутренний цикл *DO*, а внешний представить в виде:

```
      DO 3 I=1,6
      N=N+M(I,I)
    3 CONTINUE
```

Упражнение 2.8. Составить программу вычисления ряда

$$S = 1 + \frac{x}{1} + \frac{x^2}{2} + \frac{x^3}{3} + \ldots + \frac{x^n}{n} + \ldots$$

с точностью $\left|\frac{x^n}{n}\right| < \varepsilon$ для значений

x = −0,5; −0,25; 0; +0,25; +0,5 и ε = 0,0001.

Глава III

ПОДПРОГРАММЫ

В библиотеку ФОРТРАНА включены прикладные программы, наиболее часто употребляющиеся при решении многих практических задач. Однако некоторые задачи требуют использования специальных подпрограмм, не включенных в библиотеку, характерных только для определенной, конкретной задачи.

Возможности ФОРТРАНА позволяют включить в текст основных программ подпрограммы нескольких разновидностей.

Операторы - функции

Оператор-функция определяет некоторую функцию, выполняющуюся всегда, когда ссылка к этой функции появляется в другом операторе этой же программы.

Общая форма оператора:

Имя (a_1 , a_2 , a_3 , ..., a_n) = <выражение>,

где Имя - наименование оператора-функции;

$a_1, a_2, \ldots a_n$ - формальные аргументы;

выражение - любое арифметическое выражение, не содержащее индексированных переменных, любая функция, появляющаяся в этом выражении, должна быть определена в предыдущих операторах-функциях.

Формальные аргументы оператора-функции во время обращения к ней заменяются фактическими аргументами.

Пусть, например, в основной программе требуется несколько раз вычислять величину $x^2+y^2+z^2$. Эта величина (назовем ее $VECT$) есть функция трех переменных X, Y, Z

```
VECT(X,Y,Z)=X**2+Y**2+Z**2
```

Здесь имеется наименование подпрограммы ($VECT$) с ее аргументами X , Y , Z и собственно арифметическое выражение. Каждый раз, когда в основной программе будет встречаться наименование оператора-функции, произойдет ее исполнение. При этом формальные аргументы подпрограммы будут заменены фактическими.

Так, если в некотором месте основной программы потребуется вычислить выражение:

```
W=VECT(A,B,C)/(1.+VECT(D,E,F)),
```

то подпрограмма $VECT$ будет исполнена дважды, сначала с аргументами A , B , C а затем - D , E , F .

Важно отметить, что запись оператора-функции служит ее определением и сама по себе не является исполняемым оператором.

Все операторы-функции, используемые в программе, должны предшествовать первому исполнимому оператору программы. Обращение к оператору-функции возможно только в арифметическом выражении оператора присваивания. Одни и те же формальные аргументы могут быть

использованы более, чем в одном операторе-функции, а также могут быть использованы как переменные вне операторов-функций.

Тип функции в операторе-функции определяется, как и в случае переменной, либо способом неявного описания, либо явным способом. Аналогично определяются типы формальных аргументов.

В выражения операторов-функций могут быть также включены и библиотечные подпрограммы.

Например:

```
DISTAN(X,Y)=SQRT(X**2+Y**2).
```

Неправильно записанные операторы-функции:

```
TERM(3,J,K)=3*I+J**3
```

-аргументы должны быть переменными;

```
SOM(A(I),B)=A(I)/B+3
```

-аргументы должны быть неиндексированными;

```
SUBPROGRAM(A,B)=A**2+B**2
```

-имя функции превышает шесть символов;

```
3FUNC(D)=3.14*D
```

-имя функции должно начинаться с буквы;

```
ASF(A)=A+B(I)
```

-индексированная переменная в выражении;

```
BAD(A,B)=A+B+BAD(C,D)
```

-рекурсивная функция не разрешается.

П р и м е р. В основной программе требуется вычислять функцию $arccos$. Среди библиотечных функций ее нет, и поэтому можно ее вычислить через $arctg$.

Соответствующая оператор-функция:

```
ACOS(X)=ATAN(SQRT(1.-X*X)/X),
```

т.е.

$$arccos\,x = arctg\frac{\sqrt{1-x^2}}{x}$$

§2. ПОДПРОГРАММЫ ТИПА *FUNCTION*

Подпрограмма-функция используется в тех случаях, когда собственно подпрограмма состоит не из одного арифметического выражения, а требует нескольких операторов. Подпрограмма-функция является сама по себе полной программой и имеет даже собственный оператор окончания *END* .

Запись подпрограммы производится следующим образом. Сначала записывается заголовок

FUNCTION $F(X_1, X_2, \ldots, X_n)$,

где F – наименование функции;

$x_i\ (i = 1,2,\ldots, n)$ – формальные аргументы, т.е. аргументы функции F, которым еще не присвоены конкретные числовые значения.

Далее записывается группа операторов, которая вычисляет значение функции $F\ (X_1\ ,\ X_2\ ,\ \ldots,\ X_n)$ и с помощью арифметического оператора присваивает это значение наименованию функции. Заканчивается подпрограмма оператором *RETURN*, обеспечивающим возврат в основную программу, и затем оператором окончания работы подпрограммы *END* . Заметим, что все обозначения имеют силу только внутри данной подпрограммы-функции.

Тип значения, возвращаемого подпрограммой *FUNCTION*, определяется ее именем, так же как тип переменной.

Рассмотрим, например, подпрограмму вычисления факториала с наименованием *IFACT*.

П р и м е р:

```
  FUNCTION IFACT(K)
  M=K
  IFACT=1
1 IFACT=IFACT*M
  M=M-1
  IF(M)1,2,1
2 RETURN
  END
```

Обращение к подпрограмме из основной программы производится по ее наименованию, как и в случае библиотечных или арифметических подпрограмм. Например, для вычисления числа сочетаний по формуле

$C_n^m = \frac{n!}{m!(n-m)!}$

можно записать

C=IFACT(N)/(IFACT(M)*IFACT(N-M)).

Отметим, что в подпрограмме-функции в качестве как формальных, так и фактических аргументов могут выступать не только простые переменные, но и массивы. Если некоторый массив служит формальным аргументом, то он должен быть описан в операторе DIMENSION, содержащемся внутри подпрограммы-функции. Например, если подпрограмма обрабатывает массив TAB из 100 чисел, то в ней обязателен оператор

DIMENSION TAB(10Ф)

Рассмотрим еще один пример.

Составить подпрограмму вычисления суммы элементов массива из 10 целых чисел.

```
      INTEGER FUNCTION SUM(A)
      INTEGER A(1Ф)
      J=Ф
      DO 2 I=1,1Ф
      J=J+A(I)
2     CONTINUE
      SUM=J
      RETURN
      END
```

§3. ПОДПРОГРАММЫ ТИПА SUBROUTINE

Подпрограммы типа SUBROUTINE во многих отношениях сходны с подпрограммами типа FUNCTION. Но если подпрограмма-функция имеет явным результатом одну величину (результат присваивается имени процедуры), то подпрограмма типа SUBROUTINE может иметь несколько результатов, в качестве которых могут выступать как значения простых переменных, так и значения массивов.

Подпрограммы-функции вызываются в основную программу упоминанием наименования подпрограммы. Для вызова подпрограммы типа SUBROUTINE необходим специальный оператор CALL.

Первым оператором определения подпрограммы типа *SUBROUTINE* обязательно является следующая строка:

SUBROUTINE $S(P_1, P_2, \ldots, P_n)$,

где S - наименование подпрограммы;

P_i - формальные аргументы.

Далее осуществляется требуемая вычислительная процедура с использованием формальных аргументов аналогично тому, как это делалось в случае подпрограммы-функции. Однако наименованию подпрограммы не присваивается числового значения. Результаты же вычислений могут быть присвоены одному или нескольким формальным аргументам. Выход из подпрограммы, как и выход из подпрограммы-функции, осуществляется по оператору *RETURN*. В конце подпрограммы ставится оператор *END* . Отметим, что все обозначения имеют силу только внутри данной подпрограммы.

Для обращения к подпрограмме используется оператор

CALL $S(Q_1, Q_2, \ldots, Q_n)$,

где Q_i - фактические аргументы.

Примером простейшей подпрограммы типа *SUBROUTINE* может служить подпрограмма копирования, переписывающая группы чисел из одного массива в другой.

П р и м е р:

```
SUBROUTINE COPY(A,B,N)
DIMENSION A(100),B(100)
DO 1 J=1,N
1 B(J)=A(J)
RETURN
END
```

Как видно из подпрограммы, она рассчитана на работу с массивами длиной не более 100.

Обращение к этой подпрограмме из основной программы осуществляется оператором

CALL COPI(OLD,TEN,K),

по которому формальные аргументы A , B , N заменяются фактическими: *OLD* , *TEN* , K . Результатом работы подпрограммы является массив *TEN* .

Подпрограммы типа *SUBROUTINE* могут содержать внутри

себя операторы *CALL* , вызывающие другие подпрограммы типа *SUBROUTINE* и любые другие операции, в которых может содержаться обращение к подпрограммам типа *FUNCTION* , арифметическим или библиотечным подпрограммам.

Однако подпрограмма не может содержать обращение к самой себе ни непосредственно, ни через другие подпрограммы. Этому ограничению подчиняются и подпрограммы типа *FUNCTION*.

Рассмотрим пример подпрограммы вычисления суммы элементов массива из 10 целых чисел:

```
      SUBROUTINE SUM(A,V)
      INTEGER A(10),V
      J=0
      DO 2 I=1,10
      J=J+A(I)
    2 CONTINUE
      V=J
      RETURN
      END
```

Сравните эту подпрограмму с аналогичной подпрограммой типа *FUNCTION* из §2, гл. Ш.

§4. ОБРАЩЕНИЕ К БИБЛИОТЕКЕ ПРИКЛАДНЫХ ПРОГРАММ

Во многих случаях оказывается возможным вместо составления какой-либо необходимой подпрограммы обратиться к соответствующей библиотечной программе, входящей в состав математического обеспечения ЕС ЭВМ. Это значительно упростит и ускорит программирование задачи.

Обычно обращение к прикладной программе осуществляется с помощью оператора *CALL* ,так же как и в случае подпрограмм типа *SUBROUTINE*.

Рассмотрим пример обращения к несложной библиотечной программе.

"Вычисление равномерно распределенного случайного числа".

Программа вычисляет равномерно распределенные вещественные числа на интервале (0,1) и случайные целые числа между 0 и 2^{31}.

Каждое обращение к подпрограмме использует для ввода целое случайное число, выдается новое целое и вещественное случайное число.

Обращение:

```
   IX=21347
   . . . . . .
6  CALL RANDU(IX,IY,YFL)
   IX=IY
   . . . . . .
   . . . . . .
```

IX - при первом обращении к программе это целое нечетное число с девятью или меньшим числом цифр;

IY - вычисленное целое случайное число, необходимое при следующем обращении к этой программе;

YFL - вычисленное равномерно распределенное случайное число с плавающей запятой на интервале (0,1).

Полный перечень подпрограмм приведен в библиотеке прикладных программ операционной системы ДОС ЕС.

Глава IУ

СПЕЦИАЛЬНЫЕ ОПЕРАТОРЫ

Кроме описанных ранее различных операторов в языке ФОРТРАН имеются некоторые специальные операторы, позволяющие корректировать обмен информацией между подпрограммами, рационально распределять память машины между массивами чисел, выделять общие участки памяти для размещения различных данных и т.п.

§I. ОПЕРАТОР *EXTERNAL*

Оператор *EXTERNAL* объявляет в программе имена подпрограмм (*FUNCTION, SUBROUTINE* или имена других подпрограмм), которые передаются как фактические аргументы в другие подпрограммы. Иначе они будут рассматриваться как идентификаторы переменных.

Общая форма оператора:

```
EXTERNAL A,B,C,
```

где *A,B,C* – имена подпрограмм, передаваемые как фактические аргументы в другие подпрограммы.

П р и м е р.

Вызывающая программа

```
      :
EXTERNAL MULT
      :
CALL SUB(3,MULT,C)
      :
STOP
END
```

Подпрограммы

```
SUBROUTINE SUB(K,Y,Z)
IF(K)4,6,6
4 D=Y(K,Z**2)
      :
6 RETURN
  END
  FUNCTION MULT(N,S)
      :
```

Оператор *EXTERNAL* является оператором описания и поэтому должен находиться в программе до первого исполнимого оператора.

Отметим, что в операторе *EXTERNAL* можно описывать любые подпрограммы, а не только те из них, которые будут использоваться в качестве фактических аргументов.

Рассмотрим еще несколько примеров.

Если имеется подпрограмма вычисления интеграла

$$Y=\int_a^b f(x)\,dx$$

по формуле трапеций, описанная как

FUNCTION TRAP(A,B,N,F) ,

то для вычисления $\int_0^{\pi}\sin x\,dx$ по $N = 10$ точкам необходимо следующее обращение:

```
EXTERNAL SIN
. . . . . .
Y=TRAP(0.,3.14159,10,SIN)
. . . . . .
```

Если подынтегральная функция не является библиотечной, то вычисление ее должно быть оформлено как подпрограмма-функция.

Например, для вычисления

$$\int_0^1 \varphi(x)\,dx, \text{ где } \varphi(x)=\frac{\pi}{2}(1-x^2)$$

следует оформить функцию

```
FUNCTION PHI(X)
PHI=3.14159*(1-X*X)/2
. . . . . .
```

после чего вычисление интеграла по K точкам осуществляется операторами:

```
EXTERNAL PHI
Y=TRAP(0.,1.,K,PHI)
```

§2. ОПЕРАТОР *COMMON*

Операторы *COMMON* используются для определения общей области памяти для программных единиц выполняемой программы. Переменные и массивы, указанные в операторе *COMMON* в программной единице, будут занимать ту же память, что и переменные и массивы, указанные в *COMMON* в другой программной единице выполняемой программы.

Пример.

Основная программа

```
COMMON A,B,C,R(100)
REAL A,B,C
INTEGER R
.
CALL MAP(.....)
```

Подпрограмма

```
SUBROUTINE MAP(.....)
.
COMMON X,Y,Z,S(100)
REAL X,Y,Z
INTEGER S
```

Оператор *COMMON* употребляется как один из способов экономии места в памяти машины.

§3. ОПЕРАТОР *EQUIVALENCE*

Этот оператор используется, как и оператор *COMMON*, в целях экономии места в памяти машины.

Оператор *EQUIVALENCE* используется для назначения общей памяти двум или более переменным программной единицы. Эти переменные заключаются в скобки и образуют список оператора *EQUIVALENCE*.

Общая форма оператора:

EQUIVALENCE $(a_1, b_1, c_1, \ldots), \ldots, (a_n, b_n, c_n, \ldots)$

Пример 1.

```
EQUIVALENCE(X2,Y2,Z2),(ALPHA,MU)
```

Пример 2.

Если с помощью оператора *EQUIVALENCE* установлена эквивалентность места для нескольких элементов массива, то эта эквивалентность автоматически распространяется на все другие последующие и предыдущие элементы массивов:

```
EQUIVALENCE (X(3), Y(2))
            X(2), Y(1)
            X(4), Y(3)
            . . . . . .
```

Пример 3.

```
DIMENSION C(1ФФ,1ФФ),A(5Ф,5Ф),B(1ФФ)
EQUIVALENCE (C(1),A(1)),(C(25Ф1), B(1))
```

Здесь установление эквивалентности приводит к тому, что 2500 элементов массива *A* будут занимать ту же память, что и первые 2500 элементов массива *C*. Это допустимо, если эти массивы не нужны одновременно. Аналогично массив *B* будет иметь общую память с элементами 2501-2600 массива *C*.

Глава У

ОПЕРАЦИОННАЯ СИСТЕМА ЕС ЭВМ

Операционная система - это комплекс программ, предназначенных для улучшения функционирования и расширения применения ЭВМ, для автоматизации процесса подготовки программ и прохождения их на машине, увеличения производительности вычислительной системы и повышения производительности труда обслуживающего персонала. Операционные системы значительно упрощают работу программиста и оператора.

ДОС/ЕС - дисковая операционная система, обеспечивающая эксплуатацию всех моделей ЕС ЭВМ (кроме ЕС-1010 и ЕС 1021). Эта система предназначена для обеспечения работы в режиме пакетной обработки в системах с малым объемом оперативной памяти (64-256 кило-

байтов) и с ограниченным набором внешних устройств. *ДОС/ЕС* включает трансляторы с таких языков, как базисный ФОРТРАН, ФОРТРАН IУ, ПЛ/1, Кобол, РПГ.

§I. ОБРАБОТКА ПРОГРАММ ДЛЯ РЕШЕНИЯ НА ЕС ЭВМ

Основной единицей работы для операционной системы ЕС ЭВМ является задание. Каждое программное задание должно быть описано на языке управления заданиями. Язык управления заданиями является средством, связывающим программиста с операционной системой. С помощью этого языка программист может указать системе порядок выполнения задания, затребовать определенные системные средства, описать необходимые устройства ввода-вывода.

В общем случае процесс обработки задания в ЭВМ состоит из трех шагов: трансляция, редактирование и выполнение (рис. 6).

Программа, записанная на любом из языков программирования называется исходным модулем, или исходной программой. Исходная программа преобразуется соответствующим транслятором в объектный модуль. Объектный модуль - это программный модуль в промежуточном формате, общем для всех трансляторов системы. Объектные модули должны пройти еще один этап обработки - этап редактирования. В результате получается программа, готовая к выполнению. Она представляет собой совокупность программных фаз (или одну фазу). Фазу, называемую также абсолютным модулем, можно загружать в основную память для выполнения.

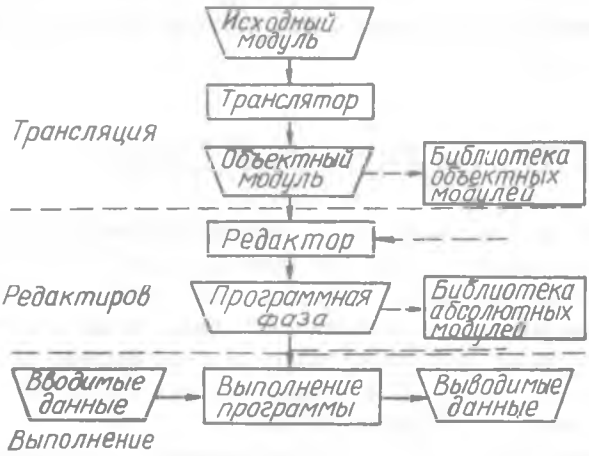
На этапе редактирования программные фазы могут быть собраны из отдельно подготовленных модулей, в них могут быть включены готовые модули, имеющиеся в библиотеке.

Язык управления заданиями состоит из управляющих операторов и, соответственно, из управляющих карт (каждый оператор перфорируется на отдельной перфокарте).

К числу управляющих операторов относятся:

JOB, OPTION, EXEC, ACTION, LINK, /, /& и др.*

Рассмотрим различные типы заданий.



Р и с. 6.

Т о л ь к о т р а н с л я ц и я

Задание состоит из одного шага: выполнения трансляции

```
//␣ JOB␣ < Имя задания >
//␣ EXEC␣ FORTRAN
< Исходная программа >
/*
/&
```

Здесь операторы исходной программы преобразуются в объектный модуль. Если объектный модуль должен быть записан во внешнюю память, нужно после первого оператора включить оператор *OPTION LINK*.

Т р а н с л я ц и я и р е д а к т и р о в а н и е

Задание состоит из двух шагов: выполнения трансляции и редактирования.

```
//␣ JOB␣ < Имя задания >
//␣ OPTION␣ CATAL
␣␣␣ PHASE␣ FIRST, *
//␣ EXEC␣ FORTRAN
< Исходная программа >
```

```
/*
//␣ EXEC␣ LNKEDT
/&
```

Здесь объектный модуль редактируется сразу после трансляции. Результирующая фаза с именем *FIRST* записывается в библиотеку абсолютных модулей для последующего выполнения.

Если объектный модуль выбирается из библиотеки объектных модулей, то оператор *//EXEC FORTRAN* нужно заменить на оператор *␣␣␣ INCLUDE␣* <Имя модуля>.

Только выполнение

Задание состоит из одного шага: выполнения фазы, которая предварительно была отредактирована и записана в памяти. Фаза выбирается из библиотеки абсолютных модулей и затем выполняется.

```
//␣JOB␣   < Имя задания >
//␣ EXEC␣ < Имя фазы  >
  < Данные >
/*
/&
```

Полная обработка

Ниже приводится минимальный обязательный набор управляющих операторов для полной обработки программы на ФОРТРАНЕ. Операторы записываются на бланке (и перфорируются), начиная с первой позиции:

```
//␣JOB␣  < Имя задания ␣ комментарии >      (I)
//␣ OPTION␣ LINK                           (2)
␣␣␣ ACTION␣ NOMAP                          (3)
//␣ EXEC␣ FORTRAN                          (4)
<Программа на ФОРТРАНЕ >
/*                                         (5)
//␣ EXEC␣ LNKEDT                           (6)
```

```
//␣ EXEC                                    (7)
        < Исходные данные >
/*                                          (8)
/&                                          (9)
```

Оператор (I) сообщает о начале очередного задания. Имя задания должно содержать от одного до восьми символов языка ФОРТРАН и начинаться обязательно с буквы. Комментарии - любой текст.

Оператор (2) сообщает системе, что объектный модуль должен быть помещен на системное запоминающее устройство для последующего редактирования.

Оператор (3) отменяет распечатку диагностических сообщений программы-редактора.

Оператор (4) начинает шаг трансляции. При этом будет использован транслятор с базисного ФОРТРАНА.

Оператор (5) указывает, что набор операторов входного языка (ФОРТРАНА) закончен.

Оператор (6) начинает шаг редактирования.

Оператор (7) начинает шаг выполнения.

После оператора *//EXEC* идут карты с исходными данными, которые вводятся по операторам ввода.

Оператор (8) завершает набор исходных данных.

Оператор (9) сообщает, что очередное задание закончено и система может переходить к следующему заданию.

Подпрограммы типа *FUNCTION* и *SUBROUTINE* представляют собой самостоятельные программные единицы. Трансляция подпрограмм и основной программы выполняется на отдельных шагах задания.

Задание на трансляцию, редактирование и выполнение программы пользователя, состоящей из основной программы и прдпрограммы типа *FUNCTION* , имеет вид:

```
// JOB PRIM ПРИМЕР ЗАДАНИЯ
// OPTION LINK
    ACTION NOMAP
// EXEC FORTRAN
      READ(1,10)M,N
   10 FORMAT(2I2)
      C=IFACT(N)/(IFACT(M)*IFACT(N-M))
      WRITE(3,20)C
   20 FORMAT(' C=',F8.3)
      STOP
      END
/*
// EXEC FORTRAN
      FUNCTION IFACT(K)
      M=K
      IFACT=1
    1 IFACT=IFACT*M
      M=M-1
      IF(M)1,2,1
    2 RETURN
      END
/*
// EXEC LNKEDT
// EXEC
 7  9
/*
/&
```

Основная программа

Подпрограмма

В этом примере основная программа вводит с перфокарт значения переменных M и N и обращается к подпрограмме *IFACT* для вычисления выражения вида $C=\frac{N!}{M!(N-M)!}$

Ниже приводится пример оформления задания на полную обработку программы с использованием подпрограммы типа *SUBROUTINE*.

Подпрограмма вычисляет сумму элементов массива *AMAS*

```
//␣JOB␣ MASSIV␣ ОПЕРАЦИИ С МАССИВОМ
//␣ OPTION␣ LINK
␣␣␣ ACTION␣ NOMAP
//␣ EXEC␣ FORTRAN
      DIMENSION A(2Ø)
      READ(1,1Ø)A
   1Ø FORMAT(1Ø F5.1)
      CALL SUM(A,2Ø,S)
      WRITE(3,2Ø)S
   2Ø FORMAT('␣ СУММА=', F8.3)
      STOP
      END
/*
//␣ EXEC␣ FORTRAN
      SUBROUTINE SUM(AMAS,N,SUMMA)
      DIMENSION AMAS(1ØØ)
      SUMMA=Ø.
      DO 15 I=1,N
   15 SUMMA=SUMMA+AMAS(I)
      RETURN
      END
/*
//␣ EXEC␣ LNKEDT
//␣ EXEC
  1  2  3  4  5  6  7  8  9  1Ø
 11  12 13 14 15 16 17 18 19  2Ø
/*
/&
```

Основная программа

Подпрограмма

§2. ДИАГНОСТИЧЕСКИЕ СООБЩЕНИЯ ТРАНСЛЯТОРА

Транслятор построен так, что синтаксический контроль исходной программы выполняется до конца, несмотря на ошибки в операторах. Различают два вида диагностических сообщений: итоговые сообщения об ошибках и сообщения об ошибках в операторах.

Итоговые сообщения об ошибках указывают на те ошибки, которые не связаны с конкретными операторами исходной программы.

Сообщения об ошибках в операторах печатаются за оператором исходной программы, в котором обнаружена ошибка. Указателем ошибки является знак денежной единицы (¤), который печатается под ошибочной позицией оператора.

Примеры:

а) в случае нецифровой метки будет выдано сообщение

```
      1A1   N=5
       ¤
Ф1)SYNTAX
```

б) отсутствует метка у оператора *FORMAT*

```
      WRITE(3,5)K
      FORMAT(I8)
        ¤
Ф1)LABEL
```

в) отсутствует третья метка в операторе *IF*

```
      IF(N-E)2,4
              ¤
Ф1)LABEL
```

§3. СООБЩЕНИЯ О СБОЯХ И ОШИБКАХ ВО ВРЕМЯ ВЫПОЛНЕНИЯ РАБОЧЕЙ ПРОГРАММЫ

Выполнение программы может быть приостановлено из-за программных сбоев или прекращено из-за ошибок.

Программные сбои возникают в результате:

1) переполнения порядка;
2) исчезновения порядка;
3) деления на нуль.

Переполнение порядка имеет место, когда результат выполнения арифметических операций равен или больше $16^{63}(\sim 7{,}2\cdot 10^{75})$.

Исчезновение порядка имеет место при получении результата, меньшего по модулю, чем $16^{-65}(\sim 5{,}4\cdot 10^{-79})$.

При программных сбоях выдается сообщение с номером 2251, после обработки сбоя выполнение программы продолжается.

Сбои обрабатываются системой следующим образом:

при исчезновении порядка в результат засылается нуль;

при переполнении порядка в результат засылается наибольшее допустимое число с плавающей запятой;

при делении на нуль результат не изменяется.

Прекращение выполнения программы происходит при возникновении ошибок. Например, попытка вычислить квадратный корень из отрицательного числа рассматривается как ошибка. При возникновении ошибок выдается сообщение, и выполнение программы прекращается.

Полный перечень сообщений об ошибках приведен в приложении I и 2 документа ЕС ЭВМ. "Операционная система ДОС/ЕС. Базисный ФОРТРАН. Руководство для программиста".

Ответы к упражнениям

I.I I) $-1.E-15$
2) $1.24E\emptyset 2$
3) $-72.64E\emptyset\emptyset$
4) $-.7E\emptyset\emptyset$
5) $.724E-\emptyset 6$
6) $32.E\emptyset 6$

I.2. I) Показатель после буквы E должен быть целым.
2) Показатель в форме E должен быть указан обязательно.
3) Показатель может иметь не более двух цифр.
4) Мантисса всегда вещественная (с точкой).

I.3. 1) $A+\frac{B}{CD}$; 2) $\frac{XYD}{C}$; 3) $\left(\frac{A+B}{C}\right)^{3,2}$

I.4. Правильная запись:

I) (X+7.)/(Y-6.)

2) ((A+B+C)/(2.*X))**3

3) (X/Y)**(R-2.)

2.1. 1) `Y=COS(ABS(3.1416*X))/(AM*SQRT(A*B)`

2) `Y=ALOG(T*T/SQRT(S-32.5))`

3) `Y=AM**Ø.75/(Y*Y+1.)`

4) `Y=EXP(Ø.3*X)*SIN(X*X)`

2.2. 1) Ø.Ø; 2) 5.; 3) 5; 4) 6.

2.3.
```
      PROGRAM SISTEM
      X=1.25
      P=.31*EXP(X)+X**5
      Q=(SIN(X))**2-X
      S=SQRT(P)+ALOG(ABS(Q-1.2E-3))
      STOP
      END
```

2.4.
```
      PROGRAM STRING
      . . . . . .
      . . . . . .
      X=(Ø.5÷3.5)
      I=X+Ø.5
      GO TO (5,7,9),I
    5 Y=SIN(X)*T*ALOG(X)
      GO TO 11
    7 Y=ALOG(X)+X
      GO TO 11
    9 Y=SQRT(SQRT(X))
   11 продолжение программы
```

2.5.
```
      PROGRAM KING
      . . . . . .
      . . . . . .
      IF(X+Y-B)2,4,4
    2 F=(X-Y/B)**2
      GO TO 6
    4 F=SQRT(Y-X/B)
    6 продолжение программы
```

2.6.

```
      PROGRAM KLIN
      A=2.375
      B=Ø.56
      X=Ø.5
I     F=ALOG((SQRT(A*X)+B)/X)
      X=X+Ø.1
      IF(X-2.5)1,1,2
2     STOP
      END
```

2.7.

```
      DIMENSION X(5ØØ)
      AMN=X(1)
      DO 4 N=2,5ØØ
      IF(ABS(X(N))-AMN)2,4,4
2     AMN=ABS(X(N))
4     CONTINUE
```

2.8.

```
      PROGRAM SIGMA
      E=.ØØØ1
      X=-Ø.5
I     S=1.
      N=Ø
2     N=N+1
      Y=X**N/N
      S=S+Y
      IF(E-ABS(Y))2,3,3
3     WRITE(3,4)S
4     FORMAT(3X,'S=',E12.6)
      X=X+.25
      IF(X-Ø.51)1,5,5
5     STOP
      END
```

Литература

1. Первин Ю.А. Основы ФОРТРАНА. М.: Наука, 1972.
2. Д.Мак-Кракен, Дорн У. Численные методы и программирование на ФОРТРАНЕ. М.: Мир, 1969.
3. Бухтияров А.М., Фролов Г.Д. Сборник задач по программированию на алгоритмических языках. М.: Наука, 1974.
4. Джермейн К. Программирование на IBM/360. М.: Мир,1973.
5. В. Дж. Калдербенк. Курс программирования на ФОРТРАНЕ-IУ. М.: Энергия, 1976.
6. Карпов В.Я. Алгоритмический язык ФОРТРАН. М.: Наука, 1976.
7. Салтыков А.И., Макаренко Г.И. Программирование на языке ФОРТРАН. М.: Наука, 1976.

С О Д Е Р Ж А Н И Е

Юрий Николаевич М а л и е в

ПРОГРАММИРОВАНИЕ НА ФОРТРАНЕ

Учебное пособие

Редактор Л. С о к о л о в а
Техн.редактор Н. К а л е н ю к
Корректор Е. А н т о н о в а

Подписано в печать 29.05.80 г. Е000397.
Формат 60х84 1/16. Бумага оберточная белая.
Оперативная печать. Усл.п.л. 4,42.Уч.-изд.л. 4,3.
Тираж 1500 экз. Заказ № 4230 Цена 15 коп.

Куйбышевский ордена Трудового Красного Знамени авиационный институт им. С.П.Королева,г.Куйбышев, ул. Молодогвардейская, 151.

Областная типография им. В.П.Мяги, г. Куйбышев, ул.Венцека,60.