Министерство высшего и среднего специального
образования Р С Ф С Р

КУЙБЫШЕВСКИЙ ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
АВИАЦИОННЫЙ ИНСТИТУТ ИМЕНИ С.П.КОРОЛЕВА

Ю.Н.Малиев,
В.В.Пшеничников

ПРОГРАММИРОВАНИЕ НА ФОРТРАНЕ

Учебное пособие
по прикладной математике

Рассмотрено и утверждено
на редакционно-издательском
совете института 4.I2.75 года

Куйбышев I 9 7 7

УДК 681.14

В учебном пособии излагаются основные методы программирования на языке Фортран.

Пособие предназначено для студентов, изучающих вопросы применения средств вычислительной техники при решении инженерных и экономических задач, а также может быть использовано научными работниками и аспирантами для выполнения на ЭЦВМ трудоемких вычислений и расчетов.

Таблиц 2. Иллюстраций 2. Библиографий 5.

ЮРИЙ НИКОЛАЕВИЧ МАЛИЕВ
ВИКТОР ВЛАДИМИРОВИЧ ПШЕНИЧНИКОВ

ПРОГРАММИРОВАНИЕ НА ФОРТРАНЕ

Учебное пособие

Редактор Э.А. Грязнова
Технический редактор Н.М. Каленюк
Корректор Л.М. Андросова

Подписано в печать 10/П - 77г.
Формат 60х84 1/16. Бумага обертка белая. Физ.п.л. 2,75.
Усл. печ. л. 2,56. Учетно-изд. л. 2,4. Тираж 1500
Цена 9 коп. Заказ 824 Темплан 1977, поз. 2333.

Куйбышевский авиационный институт им. С.П. Королева.
г. Куйбышев, ул. Молодогвардейская, 151.

Ротапринтный цех областной типографии им. Мяги,
г. Куйбышев, ул. Венцека, 60.

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 сред-

ства 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 IУ 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.

Г л а в а I

ОСНОВНЫЕ ЭЛЕМЕНТЫ ФОРТРАНА

§ I. А л ф а в и т

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.

С п е ц и а л ь н ы е с и м в о л ы:

\+ плюс

\- минус

* звездочка

/ наклонная черта (деление)

= равно (знак присвоения)

( ) скобки круглые

, запятая

. десятичная точка

' апостроф

␣ пробел

К л ю ч е в ы е с л о в а

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.

О п и с а т е л и:

INTEGER	целый
REAL	вещественный
DOUBLE PRECISION	двойная точность
DIMENSION	размерность
FUNCTION	функция
SUBROUTINE	подпрограмма

У п р а в л я ю щ и е:

GO TO	перейти к
IF	если
DO	выполнить
ASSIGN TO	присвоить
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:

1. Целые (форма I).
2. Вещественные (форма *F* и *E*).
3. Удвоенной точности (форма *D*).

Целые константы применяются для записи только целых чисел и могут состоять не более чем из I0 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 $I0^{-78} \div I0^{+75}$. Точность представления чисел составляет 7 десятичных разрядов. Для разделения целой и дробной части употребляется десятичная точка.

Применяются следующие формы записи вещественных констант (табл.I)

Таблица I

Общая форма записи		Запись на Фортране	Обычная
Фиксир. запятая форма F	$n.m$ $n.$ $.m$	2.35 423I4. .24I	2,35 423I4 0,24I
Плавающ. запятая форма E	$n.mE\pm K$ $n.E\pm K$ $.mE\pm K$	6I.24EØ5 247.E - Ø7 .253EI2	$6I,24 \cdot I0^{5}$ $247 \cdot I0^{-7}$ $0,253 \cdot I0^{I2}$

Числа с удвоенной точностью могут содержать до I7 цифр и записываются в общем виде, как

$n.mD\pm K$

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

I) $-I0^{-I5}$ 4) - 0,7
2) I24 5) 0,000000724
3) - 72,64 6) $32 \cdot I0^{+6}$

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

I) 35.4I2E+ 3. 3) .I5E + ØI4
2) 5.ØE 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 переменной (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.

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 позволяет определить их размерность и величину.

Например : *DIMENSION A(···), B(···), C(···),...,*
где A, B, C – наименование 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 4-х элементов:

$$A(1), A(2), A(3), A(4)$$

а массив B состоит из 6 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(I,I) M(2,I) M(3,I) M(I,2) M(2,2) M(3,2) M(I,3) M(2,3) M(3,3).

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
TAN(X)	$tg\,x$, 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)	$\log x$	
EXP(X)	e^x	

Отметим, что аргумент X может быть любым арифметическим выражением, численное значение которого является допустимым для данной функции.

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; U18NK;
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 необходимо 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 - 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)*.

Если записаны 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)

Примечание. 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.*

Эта операция более эффективна, потому что для выполнения функции возведения в степень требуется библиотечная подпрограмма, а при использовании операции умножения программа будет более экономичной.

3. Выражения, заключенные в круглые скобки, вычисляются в первую очередь. Если выражение, содержащее скобки, само заключено в круглые скобки, то вычисления производятся, начиная с внутренних 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·Z*.

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

4. 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:

$$4.*A*B \quad и \quad X/(-Z).$$

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

Чапример, вычисляя выражение 4 * (7/2), получим I2 (а не I4), 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:

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

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

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

$4AX-\frac{34,6\cdot10^6}{735A} \longrightarrow 4.*A*X-34.6E6/(735.*A)$

Упражнение I.3. Расшифровать следующие арифметические выражения:

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

2) $X*Y/C*D$

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

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

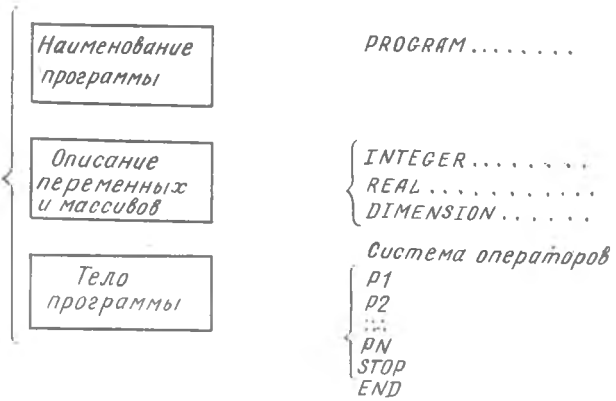
I) $\frac{X+7}{Y-6} \longrightarrow X+7./Y-6$

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

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

§ 6. Структура программ

Любая программа, записанная на языке Фортран, должна включать следующие элементы.



М е т к и

Любой из операторов может быть снабжен меткой (номером), которая должна состоять только из цифр и занимать не более 5 позиций. Например: 12, 346, 175Ø3.

Г л а в а II

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, какую 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 значе-

ние 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:

I) $y = \frac{1}{m\sqrt{ab}} cos|\pi x|$;

2) $y = \ell n\left(\frac{1}{\sqrt{s - 32{,}5}} t^2\right)$;

3) $y = \frac{m^{3/4}}{y^2 + 1}$;

4) $y = e^{0{,}3x} sin\, x^2$.

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

Т а б л и ц а 2

Тип переменной P	Арифметическое выражение Q		
	Целый	Вещественный	Удвоен.точности
Целый	присвоение	преобразование в целый	преобразование в целый
Вещественный	преобразование в вещественный	присвоение	преобразование в вещественный
Удвоенной точности	преобразование в удвоен. точность	преобразование в удвоен. точн.	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.

I) A = 2/7 ; 3) K = I9/4 + 5/4 ;
2) A=I9/4+5/4 ; 4) A = I9/I4 + 5./4.

§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 8 буквенных и цифровых символов, например, LINT2, GERON, RF5 и т.д. В конце программы ставится оператор окончания END .

Рассмотрим следующий пример.

Составить программу для вычисления одного значения функции

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

для заданных вещественных величин a , β , 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.)
END
```

Упражнение 2.3. Составить программу вычисления функции S для одного значения аргумента x = I,25, если $p = 0{,}31e^x + x^5$; $q = \sin^2 x - x$; $S = \sqrt{p} + \ln|q - 1{,}2\cdot 10^{-3}|$.

Еще один пример.

Составить программу вычисления выражения

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

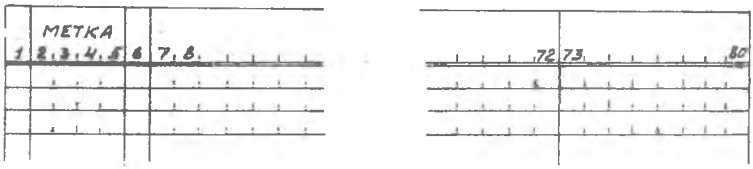
для одного значения $IND = 0{,}72\cdot 10^2$, заданного в явной форме, как вещественная величина.

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

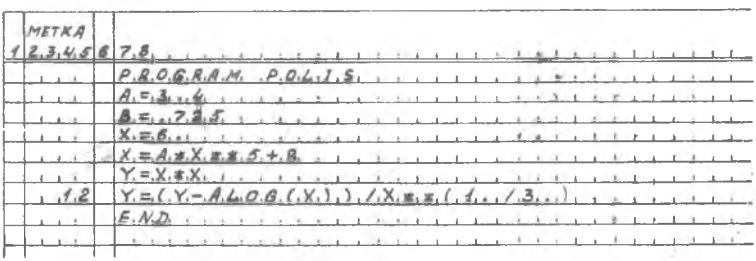
2. Каждый оператор обычно записывают на отдельной строке. Для записи операторов разрешается использовать только позиции с 7 по 72.

3. Если необходимо осуществить перенос записи оператора на следующую строку, то в 6-й 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 1-й колонке этой строки ставится буква "С".

```
C  C O M P U T A T I O N  O F  T H E  D I S C R I
C              M I N A N T
```

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

```
C  В Ы Ч И С Л Е Н И Е  Д И С К Р И М И Н А Н Т А
```

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.

§ 5. Операторы управления

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

О п е р а т о р п е р е х о д а *GO TO*

Этот оператор может употребляться в двух модификациях:

а) безусловный;

б) вычисляемый.

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 = 0.
      A = 1.
12    S = S + 1./(A*A)
      A = A+1.
      GO TO 12
      E ND
```

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 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 $y = a e^{b+1}$,
если $10 \leqslant K \leqslant 19$; то $y = a b^2 + 2$,
если $20 \leqslant K \leqslant 29$; то $y = (a \sin b)^5$.

```
PROGRAM  SWITCH
.........................
.........................
 K  =  ( 0 ÷ 29)
 I  =  K/ 1ϕ + 1
 GO TO ( 1ϕ , 12, 14), I
1Ø  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 $y = (\sin x)(t \cdot \ln x)$,
если $1{,}5 \leqslant x < 2{,}5$; то $y = \ln x + x$,
если $2{,}5 \leqslant x < 3{,}5$; то $y = \sqrt{\sqrt{x}}$

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, а α_1, α_2, α_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 достаточно сложные разветвляющиеся и циклические программы.

Рассмотрим некоторые примеры.

П р и м е р 1. Положить $x=max(a,b)$.

```
      PROGRAM KLIN
      IF(A-B)3,3,4
    3 X=B
      GO TO 7
    4 X=A
    7 END
```

П р и м е р 2. Задать x условиями:

если $y<1$, то $x=max(a,b)$,

если $y\geq 1$, то $x=1$.

```
      PROGRAM QUADR
      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 END
```

Упражнение 2.5. Составить разветвляющуюся программу вычисления функции:

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

С помощью оператора IF , как уже говорилось, можно строить различные циклические программы.

Рассмотрим примеры таких программ с простым счетчиком циклов.

Составить программу вычисления факториала числа N :

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

I Вариант

```
      IFACT=1
      K=<значение N>
1     IFACT=IFACT*K
      K=K-1          } обратный счетчик
      IF(K)2,2,1     }
2     Продолжение программы
```

II Вариант

```
      IFACT=1
      K=1
1     IFACT=IFACT*K
      K=K+1          } прямой счетчик
      IF(K-N)1,1,2   }
2     Продолжение программы
```

Оператор *IF* удобно также применять и при построении итерационных вычислительных процессов.

Рассмотрим пример программы вычисления ряда с заданной степенью точности.

Вычислить сумму членов ряда S с заданной степенью точности ε ($0 < x \leqslant 1$):

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

т.е. до выполнения условия $\frac{x^n}{n!} \leqslant \varepsilon$.

```
      PROGRAM ITERAL
      E  = < заданная степень точности ε >
      X  =  < значение x >
      A  = 1.
      B  = 1.
      S=0.0
5     Z  = A*X/B
      S  = S+Z
      B  = B+1.
      A  = Z
      IF( E-A ) 5,7,7
7     END
```

Упражнение 2.6. Составить программу вычисления квадратного корня $y=\sqrt{x}$ по рекуррентной формуле

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

с заданной степенью точности ε, т.е. до выполнения условия $|y_{i+1}-y_i|\leqslant\varepsilon$.

О п е р а т о р ц и к л а *DO*

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);

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

Операторы *DO*, 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 так далее до $N\leqslant n_2$.

Таким образом,

n_1 – нижняя константа цикла;

n_2 – верхняя константа цикла;

n_3 – приращение цикла.

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

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

Рассмотрим построение циклической программы на следующих примерах.

П р и м е р 1. Составить программу вычисления ряда

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

Фрагмент программы будет иметь вид:

```
      S = 0.0
      DO 2 N = 1,10
      R = N
      S = S + 1./R
    2 CONTINUE
```

Пример 2. Составить программу вычисления множества $\{I^3-I^2\}$:

```
      DO 5 I = 1,25
      K ( I) = I ** 2
      L (I)  = I ** 3
      M (I)  = L (I) - K ( I )
    5 CONTINUE
```

Как отмечалось, величины n_1, n_2, n_3 могут быть не только целыми константами, но и целыми переменными. Следовательно, допустимы операторы:

```
DO 15 IND = K, 50
DO 20 J   = M, LIM.
```

Упражнение 2.7. Одномерный массив A состоит из 200 элементов. Возвести в квадрат каждый нечетный элемент массива, и в третью степень - каждый четный.

Упражнение 2.8. 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.

Вложенные циклы

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

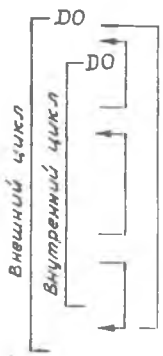
```
      S = 00
      DO 1 I = 1,12
      DO 2 K = 1,5
      S = S + AR(I,K)
    2 CONTINUE
    1 CONTINUE
```

Эта программа обеспечивает вычисление суммы элементов матрицы AR размерностью 12x5.

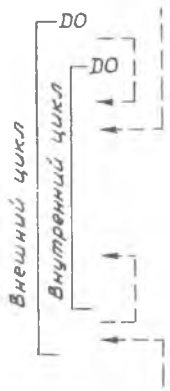
Приведенную программу можно упростить, если оба цикла будут иметь общее окончание:

```
      S = 0.0
      DO 1 I = 1,12
      DO 1 K = 1,5
      S = S + AR(I,K)
    1 CONTINUE
```

На рис. 1 схематически изображены допустимые, а на рис.2 - недопустимые передачи управления для вложенных циклов.



Р и с. 1



Р и с.2

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

Составить программу расстановки элементов массива T (I) 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 (I) 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 - I 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:

```
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
```

§ 5. Служебные операторы

Оператор *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).

Оператор *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* можно использовать 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 в начале программы можно использовать группу операторов присваивания.
Например:

```
A = 5.316
B = .0125
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 ввода:

READ (β, ∝) Список .

Аналогично, для вывода информации используется оператор вывода:

WRITE (β, ∝) 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;

∝ – 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, так называемой, автоматической индексацией.

Например, оператором *READ* (1,3 Ø) (X (I),I = 3,1Ø) будет вводиться массив X (3), X(4), ..., X (1Ø), так как здесь указан интервал изменения индекса 3 ⩽ I ⩽ 1Ø.

Двухмерный массив можно ввести, например, оператором

READ (1,16)((A(I,J), I=1,1Ø), J=1,2Ø).

Этот оператор введет элементы массива в следующем порядке:

A (1,1), A (2,1),.. . . . . , A (1Ø ,1)
A (1.2), A (2,2), ... A (1Ø ,2)
. . . .
A (1,2Ø), A (2,2Ø), ..., A (1Ø, 2Ø)

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

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)*.

Автоматическая индексация 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.

§ 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 (номер).

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

Общий вид оператора:

$$\propto FORMAT\ \underbrace{(C_1, C_2 \ldots, C_n)}_{\text{форматы}}$$

Практически используются следующие форматы:

nIw - представление целых чисел:
nFw.d, *nEw.d* - представление вещественных чисел;
nDw.d - представление чисел с удвоенной точностью;

wX - представление пробелов;
'Литерал' - представление текста.

Здесь *n* - коэффициент кратности;
w - число позиций в поле формата;
d - число цифр справа от десятичной точки.

1. Формат *I* для ввода-вывода целых чисел.

При использовании этого формата число позиций *w* должно быть не менее количества значащих цифр числа.

Например, оператор

```
READ(1,5)K
5 FORMAT(I8)
```

обеспечивает чтение с перфокарты целого восьмизначного числа K.

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

Пример.

```
      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 точки, 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 $w>d+4$.

Пример.

```
      WRITE (3,7) S
    7 FORMAT (E11.6).
```

Если число S = 65,7897 то согласно формату 7, будет выведено:
. 657897 E + 02.

4. Формат *D* аналогичен формату E и служит для ввода-вывода данных удвоенной разрядности.
Например: *WRITE (3,7) R*

7 FORMAT (D17.1Ø).

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

Рассмотрим случай ввода 20 элементов массива A, записанных на двух перфокартах по I0 величин формата *F* 6.2:

READ (1,4) A
4 FORMAT (1ØF6.2)

Если же нужно, к примеру, ввести 6 чисел (массив A) формата *F1Ø.5* с одной перфокарты и 8 чисел (массив B) формата *F* 6.4 – с другой, то используя символ " / ", получим

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

5. Формат *WX* позволяет ввести или вывести поле, состоящее из *W* пробелов.
6. Литерал позволяет вывести на печать необходимый текст в том виде, в каком он записан в операторе *FORMAT* . При этом допустимы все символы АЦПУ.

Например: 'X ='
'ЭТО ␣ ЗАГОЛОВОК'

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,I57482, то при печати получим:

␣␣*25*␣␣ *V* = ␣␣*.6349E+Ø1*␣␣␣ *S* = ␣␣*.157*

§ 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:

␣ – перевод на одну строку,
Ø – перевод на две строки,
1 – перевод на следующую страницу.
(*1* страница ≐ 63 строки)

\+ - не продвигать бумагу.

Возможности редактирования результатов с помощью первого символа 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.

Операторы

```
      WRITE (3,5)
5     FORMAT ('1 6RED')
```

позволяют перейти на новую страницу и напечатать текст:

```
6RED.
```

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

$N = 1328, \qquad A = 23{,}872, \qquad B = 0{,}7 \cdot 10^{-4}$

Это можно осуществить, например, так:

```
      WRITE (3,5) N, A, B
5     FORMAT (11X,'N =', I6/10X, 'A =', F7.3/10X,'B =', E8.2)
```

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

```
␣.....␣ N = ␣␣ 1328
␣.....␣ A = ␣ 23.872
␣.....␣ B = ␣ .70E-04
```

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 12 точках x_i, если известно, что все x_i пробиты на одной перфокарте.

x_i = 0,100; 0,109; 0,205; .....; 0,375

```
      PROGRAM TABLE
      DIMENSION X(12), F(12)
      READ (1,2) X
2     FORMAT (12 F6.3)
      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=', 12F9.4/3X, 'F(X)=', 12F9.4)
      STOP
      END
```

Решение задачи будет отпечатано в следующем виде:

```
X=0.1000   0.1090 ..... 0.3750
F(X)=2.4052  2.4422  ..... 3.5800
```

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

```
      WRITE(3,6)X(K), F(K)
    6 FORMAT(6X,'X=', F9.4,3X,'F(X)=',F9.4)
```

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

```
X =0.1000          F(X) = 2.4052
X =0.1090          F(X) = 2.4422
.................  .................
X = 0.3750         F(X) = 3.5800
```

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

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

для X = - 5, - 4, ..., +5; z = 0,01; 0,02; 0,05; 0,10; 0,20

```
      PROGRAM TABUL
      DIMENSION Z(5)
      READ(1,1)(Z(I), I=1,5)
1     FORMAT(5F6.2)
      X = - 5.
2     WRITE(3,3)X
3     FORMAT(4X, 'X=', F3.0)
      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
```

Упражнение 2.9. 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 $\left|\frac{x^n}{n}\right| \leqslant \varepsilon$ для значений X = - 0,5; - 0,25 ;

0; + 0,25; + 0,5 и ε = 0,0001.

Г л а в а III

ПОДПРОГРАММЫ

В библиотеку Фортрана включены стандартные подпрограммы, 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 включать в текст основных программ подпрограммы нескольких разновидностей.

§ 1. Подпрограммы арифметического типа

Арифметическая подпрограмма подобно арифметическому оператору вычисляет одно единственное арифметическое выражение. При этом формальные параметры (аргументы) арифметического выражения могут при каждом исполнении подпрограммы заменяться фактическими.

Пусть, например, в основной программе требуется несколько раз вычислять сумму квадратов координат точки в пространстве. Эта величина (назовем ее VECT) есть функция трех переменных X, Y и Z :

$$VECT(X,Y,Z) = X**2 + Y**2 + Z**2 . \qquad (3.1)$$

Здесь имеется наименование подпрограммы (VECT) с ее параметрами X, Y, Z и собственно арифметическое выражение. Каждый раз, когда в основной программе будет встречаться наименование арифметической подпрограммы, произойдет ее исполнение. При этом формальные параметры 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 будет исполнена дважды: сначала с параметрами A,B,C, а затем - D , E , F .

Важно отметить при этом, что запись подпрограммы (3.1) служит по существу определением функции подпрограммы и сама по себе не является исполняемым оператором. Определение подпрограммы записывается

в основной программе лишь один раз, но исполняется она каждый раз, когда ее наименование вместе с параметрами встречается в выражениях. Запись арифметических подпрограмм обычно производится в начале основных программ, перед исполняемыми операторами. Наименование подпрограммам дается произвольно.

В выражениях арифметических подпрограмм могут быть также включены и библиотечные подпрограммы. Например, арифметическая подпрограмма вычисления расстояния точки с координатами X и Y от начала координат может быть записана

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

§ 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 параметры, 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, также как тип переменной.

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

Пример.

```
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 параметров могут выступать не только простые переменные, но и массивы. Если некоторый массив служит формальным параметром, 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 (100).
```

§ 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(P1, P2, ... Pn),
```

где S - наименование подпрограммы;

P_i - формальные параметры.

Далее осуществляется требуемая вычислительная процедура с использованием формальных параметров 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 параметрам. 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 параметры.

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

Пример.

```
      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 COPY(OLD, TEN, K),

по которому формальные параметры *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*.

Глава IУ

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 килобайтов) и с ограниченным набором внешних устройств. ДОС ЕС включает трансляторы с таких языков как базисный Фортран, Фортран IУ, *PL/1* ,Кобол, РПГ.

§1. Оформление программы для решения на ЕС ЭВМ

Основной единицей работы, выполняемой вычислительной системой, является задание. Задание включает выполнение одной или нескольких связанных программ.

Каждое задание должно быть описано на языке управления заданиями, который не является языком программирования и не может быть использован для написания программ. 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.

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

Для решения задачи на Фортране необходимо выполнить три этапа (шага) обработки:
I) трансляция; 2) редактирование; 3) выполнение.

Трансляция - это процесс преобразования набора операторов языка Фортран в группу команд на машинном языке. Результат, полученный после трансляции, называется объектным модулем.

Редактирование - получение программы в выполнимой форме (абсолютный модуль).

Выполнение - это процесс загрузки абсолютного модуля в оперативную память и выполнение программы.

Ниже приводится минимальный обязательный набор управляющих операторов для полной обработки программы на Фортране. Операторы перфорируются, начиная с первой позиции:

```
//␣JOB␣< имя задания ␣ комментарии >          (I)
//␣OPTION␣LINK                                (2)
//␣EXEC␣FORTRAN                               (3)
  < Программа на Фортране >
/*                                            (4)
                                              (5)
//␣EXEC␣LNKEDT                                (6)
//␣EXEC
  < исходные данные >
/*                                            (7)
/&                                            (8)
```

Оператор (I) - сообщает о начале очередного задания.
Имя задания должно содержать от I-го до 8-ми символов языка Фортран и начинаться обязательно с буквы. Комментарии - любой текст.

Оператор (8) - сообщает, что очередное задание закончено и система может переходить к следующему заданию.

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

Оператор (3) - начинает шаг трансляции. При этом будет использован транслятор с базисного Фортрана.

Оператор (4) - указывает, что набор операторов входного языка (Фортрана) закончен.

Оператор (5) - начинает шаг редактирования.

Оператор (6) - начинает шаг выполнения.

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

Оператор (7) завершает набор исходных данных.

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 подпрограммы типа FUNCTION , имеет вид:

```
//␣JOB␣PRIM␣          ПРИМЕР ЗАДАНИЯ
//␣OPTION␣LINK
//␣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*.

ОПЕРАЦИИ С МАССИВОМ

```
// JOB MASSIW
// OPTION LINK
// EXEC FORTRAN
        DIMENSION A(20)
        READ(1,10) A
     10 FORMAT(10F6.3)
        CALL SUM (A,20,S)
        WRITE (3,20)S
     20 FORMAT (' СУММА =', F8.3)
        STOP
        END
/*
// EXEC FORTRAN
        SUBROUTINE SUM(AMAS,N, SUMMA)
        DIMENSION AMAS (100)
        SUMMA = 0.
        DO 15 I=1,N
     15 SUMMA= SUMMA + AMAS(I)
        RETURN
        END
/*
// EXEC LNKEDT
// EXEC
 1  2  3  4  5  6  7  8  9  10
11  12 13 14 15 16 17 18 19  20
/*
/&
```

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 значения об ошибках указывают на те ошибки, которые не связаны с конкретными операторами исходной программы. Они выдаются в конце распечатки за 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   I=5
Ø1) SYNTAX    ¤
```

б) при отсутствии метки у оператора FORMAT

```
          107  WRITE(3,5)I
               FORMAT(I2Ø)
Ø1) LABEL      ¤
```

в) при отсутствии третьей метки в операторе IF

```
          IF(1ØØ-I)7,4
Ø1) 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 225I, 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 1 и 2 документа ЕС ЭВМ " Операционная система ДОС/ЕС. Базисный Фортран. Руководство для программиста".

О т в е т ы к у п р а ж н е н и я м

1.1. 1) $-1.E-15$ 4) $-.7E\emptyset\emptyset$

2) $1.24\ E\emptyset 2$ 5) $.724\ E-\emptyset 6$

3) $-72.64E\emptyset\emptyset$ 6) $32.\ E\emptyset 6$

1.2. 1) 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).

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

1.4. Правильная запись:

1) `(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))
END
```

```
2.4.  PROGRAM STRING
      ....................
      ....................
   X  = (ϕ. 5 ÷  3.5)
   I  =   X + ϕ .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 FELIX
   X = < значение x >
   E = < заданная степень точности ε >
   Z = < начальное приближение y0 >
 3 Y =.5 (X/Z + Z )
   IF (E-ABS(Y-Z))4,5,5
 4 Z = Y
   GO TO 3
 5 END

2.7 DO 3 J = 2,2ϕϕ,2
    A(J-1) = A(J-1)**2
    A(J)  = A(J) ** 3
 3 CONTINUE

2.8 AMN =X(1)
    DO 4 N = 2,5ϕϕ
    IF(X(N)) 1,2,2
```

```
    1 X(N) = -X(N)
    2 IF(X(N)-AMN)3,4,4
    3 AMN = X
    4 CONTINUE
```

2.9.

```
      PROGRAM SIGMA
      E = .ØØØ1
      X = -Ø.5
    1 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

I. Первин Ю.А. Основы Фортрана. М., " Наука", I972.

2. Мак-Кракен Д., Дорн У. Численные методы и программирование на Фортране. М., " Мир", I969.

3. Бухтияров А.М., Фролов Г.Д. Сборник задач по программированию на алгоритмических языках. М., " Наука", I974.

4. Джермейн К. Программирование на IBM/360. М., " Мир", I973.

5. Калдербенк В.Дж. Курс программирования на Фортране - IУ. М., " Энергия", I976.

6. Карпов В.Я. Алгоритмический язык ФОРТРАН. М., "Наука", I976.

7. Салтыков А.И., Макаренко Г.И. Программирование на языке ФОРТРАН. М., " Наука", I976

О г л а в л е н и е