Министерство высшего и среднего специального
образования РСФСР

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

ВВЕДЕНИЕ В ФОРТРАН ДЛЯ СМ ЭВМ

Методические указания
к лабораторным работам

Куйбышев 1990

Составитель И.С.К а с а т к и н а

УДК 681.3.06

Введение в ФОРТРАН для СМ ЭВМ: Метод.указания к лаб.работам;Куйбышев.авиац.ин-т./Сост.И.С.Касаткина; Куйбышев, 1990. 78 с.

Содержатся начальные сведения по языку ФОРТРАН, необходимые студентам при подготовке программ к циклу лабораторных работ. Приведены примеры программ.

Предназначены для студентов I-4 факультетов. Составлены на кафедре "Программное обеспечение вычислительных систем".

Печатается по решению редакционно-издательского совета Куйбышевского ордена Трудового Красного Знамени авиационного института имени академика С.П.Королева

Рецензенты: В.К.С е м е н ы ч е в, О.С.И в а н о в а

ВВЕДЕНИЕ В ФОРТРАН ДЛЯ СМ ЭВМ

Редактор Е.Д.А н т о н о в а
Техн.редактор Н.М.К а л е н ю к
Корректор Н.С.К у п р и я н о в а

Подписано в печать 24.I2.90. Формат 60x84 I/I6.
Бумага оберточная. Печать офсетная. Усл.п.л.I,62.
Усл.кр.-отт. I,7. Уч.-изд.л. I,58. Т.I00 экз.
Заказ № 67. Бесплатно.

Куйбышевский ордена Трудового Красного Знамени авиационный институт имени академика С.П.Королева.
443086 г.Куйбышев, Московское шоссе, 34.

Участок оперативной полиграфии КуАИ. 443001 г.Куйбышев, ул. Ульяновская, I8.

I. ОСНОВНЫЕ ЭЛЕМЕНТЫ ФОРТРАНА

I.I. Алфавит

ФОРТРАН построен на базе английского языка, поэтому в нем используются 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.

Кроме того ФОРТРАН использует:

десять арабских цифр: 0, I, 2, 3, 4, 5, 6, 7, 8, 9;

специальные символы:

+ плюс
- минус
* звездочка (знак умножения)
/ наклонная черта (знак деления)
= равно (знак присвоения)
( ) скобки круглые
, запятая
. десятичная точка
' апостроф (кавычка)
␣ пробел
$ денежный символ
& амперсанд (коммерческое "И")

I.2. Знаки операций отношения

больше	*.GT.*
больше или равно	*.GE.*
меньше	*.LT.*
меньше или равно	*.LE*
равно	*.EQ.*
не равно	*.NE*

ЗНАКИ логических операторов:

не .NOT.
и .AND.
или .OR.

I.3. Числовые константы

Числовые константы могут быть следующих видов:
целые (форма I);
вещественные (форма *F* или *E*);
удвоенной точности (форма *D*).

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

Максимальное число: +32767.
Минимальное число: -32767.

Примеры: 27
0
-I38
+69

Вещественные константы применяются для записи рациональных чисел из следующей допустимой области значений:

$$\approx 2{,}9 \cdot 10^{-38} < |X| < \approx 1{,}7 \cdot 10^{38}.$$

Точность представления чисел составляет приблизительно 7 десятичных значащих цифр. 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 запись
Фиксированная запятая	*n.m*	3.25	3,25
	n.	I314.	I314
	.m	.017	0,017
Плавающая запятая	*n.mE±k*	91.42E05	$91{,}42 \cdot 10^{5}$
	n.E±k	314.E-2	$314 \cdot 10^{-2}$
	.mE±k	.552E21	$0{,}552 \cdot 10^{2}$

Вещественные константы с удвоенной точностью могут содержать до 17 цифр и записываются аналогично вещественной константе с плавающей запятой, только вместо буквы E указывается D .

1.4. Простая переменная

Для наименования (идентификации) переменных применяется последовательность букв и цифр, начинающаяся с буквы. Число символов в идентификаторе (имени переменной) не должно превышать шести.

Пример: A, T124, MAX90, PRIMER, C31GAM

Описание переменных. В ФОРТРАНЕ существует два способа описания переменных (объявления типа переменных):

неявное определение типа переменной (по умолчанию);

явное определение типа переменной.

Если имя переменной не начинается с одной из букв I, J, K, L, M, N, то по умолчанию это имя имеет вещественный тип.

Если имя переменной начинается с одной из букв I, J, K, L, M, N, то по умолчанию это имя имеет целый тип.

Например:

MAMA – целый тип.

F12 – вещественный тип.

При явном определении типа переменной используются ключевые слова-описатели: (INTEGER, REAL, LOGICAL, COMPLEX).

Например:
```
INTEGER Z, NAME1, X11
REAL I, M, A21B, TIME
LOGICAL B, C, D90
COMPLEX I10, J20
```

Такие описания приводятся в начале программы.

1.5. 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 или несколькими индексами, которые пишутся в скобках после наименования массива.

Например:

A (5), M (I, 2, 5), Ж IØ (I, 2, 3, 4, 5, 6, 7).

Количество индексов, разделенных запятыми, не может быть более семи.

В качестве индексов в выражениях можно использовать не только натуральные числа, но и переменные "целого" типа, которым в ходе выполнения программы должны быть при воены положительные значения, а также выражения "целого" типа.

Например: A(K), M(5, I+2, J*I), C(I+1, J1Ø+I*K).

Упорядоченная последовательность переменных с индексами, объединенная общим наименованием, называется м а с с и в о м. Массивы обязательно должны быть описаны в начале программы.

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

Для явного определения типа элементов массива можно воспользоваться ключевыми словами-описателями (INTEGER, REAL, LOGICAL, COMPLEX).

Например: REAL A(5), IT(3,4,2)
INTEGER J1Ø(3,2), B(7).

С помощью этих описателей явно описан массив А, состоящий из пяти элементов вещественного типа, массив IT, состоящий из 24 элементов вещественного типа, массив J1Ø (3,2), состоящий из 6 элементов целого типа, и массив В, состоящий из 7 элементов целого типа.

Расположение массивов в памяти. Элементы массива записываются в памяти таким образом, что значение первого индекса (левого) меняется наиболее быстро.

Двумерные массивы размещаются в памяти по столбцам.

Например: INTEGER I (3,2) разместится в памяти следующим образом:

I (1,1), I (2,1), I (3,1), I (1,2), I (2,2), I (3,2).

2. ОПЕРАТОР ПРИСВОЕНИЯ

2.1. Арифметические выражения

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

** возведение в степень;
* умножение;
/ деление;
+ сложение;
- вычитание.

Арифметическими выражениями называются:

а) числовые константы,
например: 52, - .01, 3.14E - 5;

б) переменные величины,
например: A, X(1,2), I, RON, Z15K;

в) обращения к элементарным математическим функциям,
например: SIN(X); EXP(Z);

г) обращения к подпрограммам типа FUNCTION;

д) числовые константы, переменные величины и функции, объединенные знаками арифметических операций и круглыми скобками,
например: A+(B2*X-3.5)**3-SIN(X/4.8).

При построении арифметических выражений надо соблюдать следующие правила.

1. 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 встречается подряд несколько операций умножения и деления, то действия выполняются подряд слева направо.

Аналогично выполняются операции сложения и вычитания.

Например:
выражение A/B*C означает $\frac{AC}{B}$, но не $\frac{A}{BC}$;
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: $B**C$

$A**(B**C)$

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

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

например $4ab$ 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 "I2" (а не "I4"), т.к. частное от деления "7" на "2" будет равно "3" (а не "3,5" как в обычной арифметике).

2.2. Указатели функций

С помощью специальных ключевых слов-указателей функции можно обращаться к алгоритму вычисления функции. ФОРТРАН имеет целый набор различных указателей функций.

Рассмотрим основные:

ALOG(X)	$\ln x$
ALOG10(X)	$\lg x$
EXP(X)	e^x
SQRT(X)	$\sqrt{x}$
SIN(X)	$\sin(x)$
COS(X)	$\cos(x)$
ATAN(X)	$arctg(x)$
ABS(X)	$\lvert x \rvert$

В качестве "X" могут использоваться имена переменных, константы, указатели функций, арифметические выражения.

2.3. Логические выражения

Л о г и ч е с к о е в ы р а ж е н и е - запись, состоящая из логических операндов, знаков логических операций и скобок.

Л о г и ч е с к и е о п е р а н д ы - отношения, а также константы, переменные и указатели функций логического типа.

Отношение имеет вид aRb,

где a и b арифметические выражения целого или вещественного типа;

R - знак операции отношения (см. п.I.2).

(.GT., .GE., .LT., .LE., .EQ., .NE.)

Например: (A+B)**6/X .GT. 10.5

X .NE. Y+1

Результат операции "Отношение" относится к логическому типу и принимает значения .TRUE и .FALSE

.TRUE. - если соответствующее отношение удовлетворяется для входящих в него арифметических выражений,

.FALSE. - в противном случае.

Знаками логических операций являются символы .NOT., .AND., .OR.

В табл. 2 приведены правила выполнения логических операций.

Т а б л и ц а 2

A	.TRUE.	.TRUE.	.FALSE.	.FALSE.
B	.TRUE.	.FALSE.	.TRUE.	.FALSE.
.NOT. A	.FALSE.	.FALSE.	.TRUE.	.TRUE.
A.AND. B	.TRUE.	.FALSE.	.FALSE.	.FALSE.
A.OR. B	.TRUE.	.TRUE.	.TRUE.	.FALSE.

При вычислении значений логических выражений установлен следующий порядок и уровни старшинства операций.

1. Вычисление арифметических выражений.
2. Вычисление отношений.
3. Отрицание.
4. Логическое умножение.
5. Логическое сложение.

Операции выполняются слева направо. Выражение в скобках вычисляется до того, как его значение будет использовано как операнд.

2.4. Арифметический оператор присваивания

К о м а н д н ы е о п е р а т о р ы (выполняемые 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.

Пусть а - имя переменной или элемент массива целого, вещественного или комплексного типа; в - арифметическое выражение. Тогда общий вид арифметического оператора присваивания

а = в

При выполнении оператора присваивания (=) сначала вычисляется арифметическое выражение "в", а затем осуществляется присваивание его значения переменной "а". При этом знак "=" играет роль символа присваивания. Данный оператор может использоваться для присваивания переменным конкретных числовых значений, а также значений других переменных и арифметических выражений.

Например:
```
A = 5.
C = A
X = A**N  (N - 2).
```

Формула $z = \frac{a^5}{y}\cos^2 x$ будет записана так: `Z = A**5/Y*COS(X)**2.`

С помощью оператора арифметического присваивания можно преобразовывать значение арифметического выражения "в" к типу переменной "а" по следующим правилам.

I. Если "а" - имя переменной целого типа, а "в" - выражение вещественного типа, то "а" присваивается целая часть значения "в".

Например: I = 3.95 ⟶ I = 3.

2. Если "а" - имя переменной целого типа, а "в" - комплексного типа, то в качестве "а" берется целая часть действительной части значения "в".

Например: I = (I0.6, II.9) → I = I0.

3. Если "а" - имя переменной вещественного типа, а "в" - целого типа, то в качестве "а" берется значение "в", преобразованное к вещественному типу.

Например: F = I0 } изменяется форма представления значений I0 и переменной I (преобразуется к вещественной форме представления)
E = I }

4. Если "а" - имя переменной вещественного типа, а "в" - комплексного типа, то в качестве "а" берется значение действительной компоненты "в".

Например: D = (I0.5, 7.I) → D = I0.5.

5. Если "а" - имя переменной комплексного типа, а "в" - целого или вещественного типа, то в качестве значения действительной компоненты "а" берется значение "в" (если нужно, то производится преобразование типов), а в качестве мнимой компоненты "а" берется "0".

Например: A = I0 → A = (I0,0)
B = I0.5 → B = (I0.5,0).

2.5. Простейшие программы

Начинается программа операторами описания. В конце программы ставится оператор конца *END*.

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.

```
REAL A,B,X,Y
A = 3.4
B =.725
X = 6.
C = A * X **5 + B
D = C * C
```

```
      V=(D-ALOG(C))/C**(1./3.)
      STOP
      END
```

2.6. 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 (рис. 1).

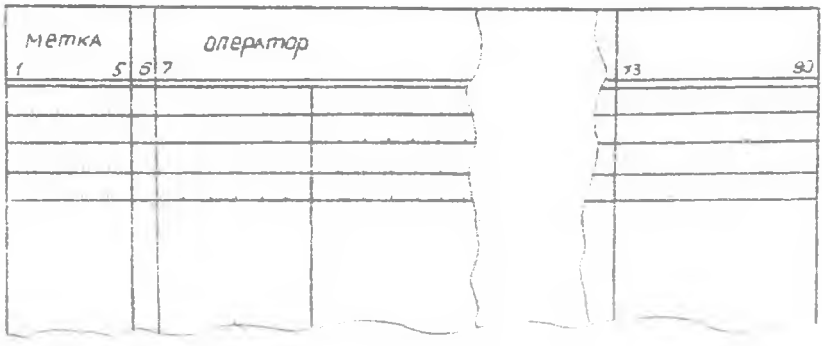


Рис. 1

При записи программы на бланке надо соблюдать следующие правила.

1. В каждой позиции записывается не более одного символа (буква, цифра, точка, запятая, скобка и т.д.), 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 шестую позицию этой строки надо поставить любой символ ФОРТРАНа, отличный от нуля и пробела.

4. Метка оператора записывается только в 1-5 позициях.

5. Один оператор с продолжением может занимать не более 20 строк.

6. На бланке в любом месте можно писать комментарии, тогда в первой колонке этой строки ставится буква С. Комментарий может содержать текст на русском языке.

Например:

```
C  В Ы Ч И С Л Е Н И Е   Ф У Н К Ц И И
```

Бланки ФОРТРАНа предназначены, главным образом, для удобства перфорирования программы. Одна строка бланка соответствует одной 80-колонной перфокарте и одной строке на экране дисплея, при этом каждая позиция бланка соответствует одной колонке перфокарты и одной позиции на строке терминала.

3. ОПЕРАТОРЫ УПРАВЛЕНИЯ

Операторы управления имеют несколько разновидностей. Их использование позволяет строить сложные программы с разветвлениями.

3.1. Метки

Любой из выполняемых операторов языка ФОРТРАН может быть снабжен меткой, которая состоит из цифр и занимает не более 5 позиций. В программе не должно быть одинаковых меток.

3.2. Оператор безусловного перехода

Оператор имеет вид GOTO α, где α - метка (числовой набор от 1 до 5 позиций). При выполнении этого оператора происходит передача управления (переход) к оператору с меткой α .

Например:

```
      REAL S,A
      S = 0
      A = 1.
10    S = S-1./(A*A)
      A = A+1.
      GO TO 10
      STOP
      END
```

При исполнении этой программы будет вычисляться бесконечный ряд $S = 1/1^2 + 1/2^2 + 1/3^2 + \ldots$

3.3. Вычисляемый оператор

Оператор имеет вид $GOTO(\alpha_1, \alpha_2, \ldots, \alpha_n), i$.

Здесь в скобках помещены метки α_i операторов, а i -имя целочисленной переменной (целая переменная должна принимать значения из диапазона номеров позиций меток). Количество меток в круглых скобках обычно не ограничивается. Более того, одна и та же метка может появляться в списке несколько раз.

Например, допустим, оператор

$GOTO$ (40, 60, 80, 40, 60, 80), J

Вычисляемый оператор $GOTO$ работает как переключатель, передавая управление на ту метку (и соответственно оператор), порядковая позиция которой указывается в данный момент значением переменной.

Например, если в приведенном выше примере $J=3$, то управление передается на метку 80 и т.д.

Если значение переменной i находится из значений диапазона позиций номеров меток, то никакого перехода не происходит, и выполняется следующий за $GOTO$ оператор.

3.4. Оператор условного перехода (арифметический)

Оператор условного перехода арифметического типа имеет вид

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

где Q - некоторое арифметическое выражение;

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

При выполнении этого оператора вначале вычисляется значение Q , а затем передается управление на одну из меток 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 .

Следует помнить, что все три метки обязательны, но не обязательно различны. Следующий за арифметическим IF оператор обязательно должен иметь метку (иначе он никогда не будет выполнен).

3.5. Логический оператор *IF*
(условный логический оператор перехода)

Логический оператор *IF* имеет вид

IF (логическое выражение) выполняемый оператор. В качестве выполняемого оператора может использоваться любой выполняемый оператор ФОРТРАНа за исключением логического *IF* и оператора цикла *DO* .

При выполнении логического оператора *IF* вначале вычисляется логическое выражение. Если полученное значение логического выражения "ложь" (*.FALSE.*), то выполняется оператор, следующий за логическим оператором *IF* .

Если значение логического выражения "истина" (*.TRUE.*), то предварительно выполняется оператор, указанный в условном *IF* . Если этот оператор, в свою очередь, является оператором безусловного или условного арифметического перехода,
то управление передается указанной в них метке. Если нет, то после его выполнения управление передается оператору, следующему непосредственно за оператором условного логического перехода.

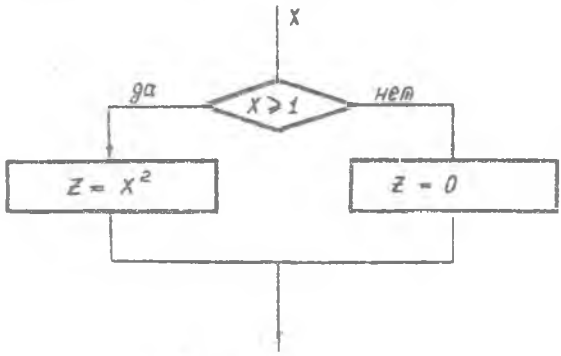
3.6. Разветвляющиеся программы

При решении многих задач ход вычислительного процесса зависит от значений переменных, входящих в расчетные формулы, т.е. в зависимости от конкретных величин переменных задачи на определенном этапе расчетов дальнейшие вычисления могут производиться по одному из 2-х или нескольких направлений. Также вычислительные процессы называются разветвляющимися, и для них составляются разветвляющиеся программы.

Пример. Составить программу вычисления z в зависимости от условия:

$$z = \begin{cases} x^2, & при\ x \geq 1, \\ 0, & при\ x < 1. \end{cases}$$

Схема алгоритма приведена на рис. 2.



Р и с. 2

Стандартная реализация ветвления на языке ФОРТРАН имеет вид:

```
      IF (условие) GO TO M1
      Операторы ветви "нет"
      GO TO M2
M1    Операторы ветви "да"
M2    CONTINUE
```

Таким образом, получим следующую программу:

```
      REAL  X, Z
      . . .
      IF (X.GE.1) GO TO 1
      Z = 0
      GO TO 2
1     Z = X*X
2     CONTINUE
      . . .
```

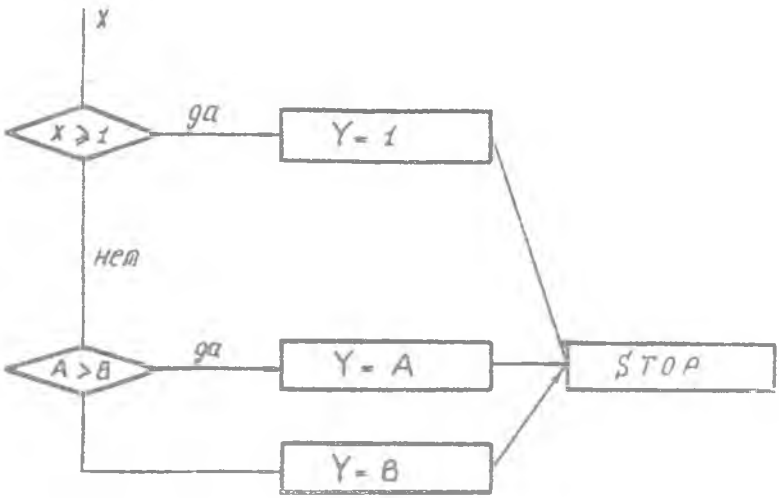
В частных случаях иногда удобнее отступать от стандартной реализации, чтобы получить более компактную программу. Приведем примеры таких разветвляющихся программ.

Пример 1. Составить программу определения при условиях:

если X < I, то У = макс (A, B);

если X ⩾ I, то У = 1

Блок-схема решения задачи приведена на рис. 3.



Р и с. 3

Составим две версии программы: с использованием оператора условного арифметического перехода и логического оператора IF :

```
      REAL  Y, X, A, B
      ...
      IF (X-1.) 10, 20, 20
10    IF (A-B) 30, 30, 40
30    Y = B
      GO TO 50
40    Y = A
      GO TO 50
20    Y = 1.
50    STOP
      END
```

```
      REAL  Y, X, A, B
      ...
      IF (X .GE. 1.) Y = 1.
      IF (X .LT. 1.) GO TO 10
      ...
      STOP
10    Y = B
      IF (A .GT. B) Y = A
      STOP
      END
```

Пример 2. Составить программу вычисления величины $x=f(y)-\sqrt{z}$, где $y=z+2$, при

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

Блок-схема программы представлена на рис. 4.

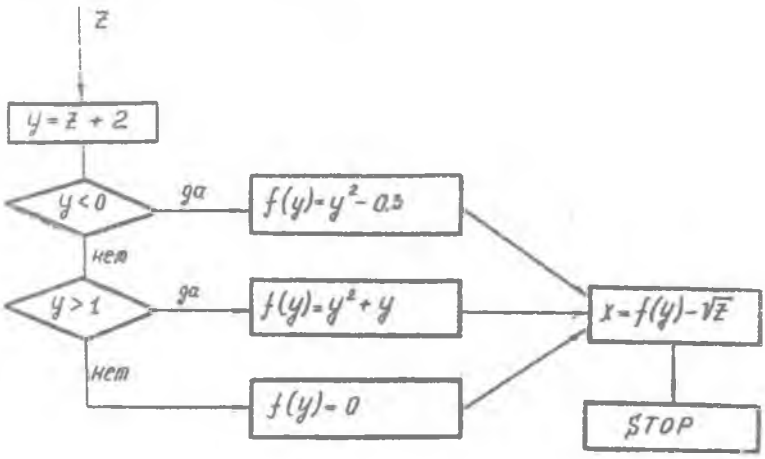


Рис. 4

Программы имеют вид

```
      REAL X,Y,Z,F
      . . .
      Y=Z+2.
      IF(Y) 10,20,20
10    F=Y*Y-0.3
      GOTO 50
      IF(Y-1.) 30  30,40
30    F=0.
      GO TO 50
40    F=Y*Y+Y
50    X=F-SQRT(Z)
      STOP
      END
```

```
      REAL X,Y,Z,F
      . . .
      Y=Z+2.
      F=0.
      IF(Y.LT.0.)F=Y*Y-0.3
      GO TO 10
      IF(Y.GT.1.)F=Y*Y+Y
10    X=F-SQRT(Z)
      STOP
      END
```

4. БЕСФОРМАТНЫЙ ВВОД-ВЫВОД

Этот вид ввода-вывода, как правило, используют при отладке программы. Он удобен тем, что вид ввода-вывода так же, как и расположение числовой информации на носителе, не нужно подробно и точно описывать, рассчитывать.

Ввод: *ACCEPT* * , список

Например: *ACCEPT* *, *A, B, K*

Этот оператор останавливает выполнение программы, при этом следует ввести с терминала числовые значения переменных, указанных в списке. Числа вводятся через запятую, после окончания ввода нажимается клавиша *<CR>* или равнозначная ей клавиша.

Ввод: *TYPE* *, список

Например: *TYPE* *, *A, B*

По этому оператору будут выданы 2 числа на экран дисплея. Можно оформить вывод следующим образом:

TYPE *, *'A='*, *A*, *'B='*, *B* .

Тогда на экране будет

A = число, B = число.

Примечание: поскольку выполнение программы останавливается при использовании оператора *ACCEPT*, рекомендуется, чтобы знать, что остановка произошла именно по этой причине, перед оператором *ACCEPT* располагать оператор *TYPE* для подсказки.

Например: 1. *TYPE* *, "введите 2 числа A и B"
ACCEPT *, *A, B*

2. *REAL A(20)*
INTEGER I

TYPE *, "введите 20 элементов массива A"
ACCEPT *, *A*

Для вывода также можно использовать оператор *PRINT* , который позволяет выводить результаты как на экран терминала, так и на печать.

Например: *PRINT* *, *'A='*, *A*, *'B='*, *B*.

5. ЦИКЛИЧЕСКИЕ ПРОГРАММЫ

Большинство численных методов решения задач сводится к многократному повторению некоторой группы операций, причем, обычно задается или число повторений, или условие их окончания. Такие вычислительные процессы называются **циклическими**, а многократно повторяющиеся участки программ **циклами**.

В зависимости от способа организации и назначения циклические процессы можно разделить на простые, итерационные, кратные (вложенные циклы.

5.1. Простой цикл

В тех случаях, когда требуется повторить определенную группу операторов заданное число раз, можно использовать различные схемы построения циклов. Рассмотрим некоторые из них.

Пример. Составить программу вычисления факториала числа N :

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

Рассмотрим фрагмент программы с прямым счетом.

```
      ...                          ...
      IFACT=1                      IFACT=1
      K=1                          K=1
10    IFACT=IFACT*K          10    IFACT=IFACT*K   }
      K=K+1                        K=K+1           } ЦИКЛ
      IF(K-N)10,10,20              IF(K.LE.N) GO TO 10 }
20    ...                          ...
```

прямой счетчик

Рассмотрим фрагмент программы с обратным счетом.

```
      ...                          ...
      IFACT=1                      IFACT=1
      K=N                          K=N
10    IFACT=IFACT*K          10    IFACT=IFACT*K   }
      K=K-1                        K=K-1           } ЦИКЛ
      IF(K)20,20,10                IF(K.GT.0)GO TO 10 }
20
```

обратн. счетчик

5.2. Итерационный цикл

Итерационные процессы характеризуются тем, что искомая величина У 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.

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

где ε – заданная степень точности.

Модуль определяется здесь в связи с тем, что получаемые разности $y_{i+1} - y_i$ могут оказаться отрицательными или знакопеременными. В итерационных процессах количество повторений циклов заранее неизвестно, поэтому вместо счетчиков циклов в конце каждого цикла нужно предусмотреть проверку окончания по вышеприведенному условию.

В некоторых случаях вычисления целесообразно производить не с абсолютной, а с относительной степенью точности. Тогда условие окончания вычислений имеет вид

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

Пример: 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 $|y_{i+1} - y_i| \leq \varepsilon$.

Обозначим $Y1 = y_i$, $Y2 = y_{i+1}$.

За начальное (грубое) приближение примем значение $Y1 = y_0$.

Программа имеет вид .

```
REAL A, E, Y1, Y2, DELTA
TYPE *, ' введите значения A, B, Y1 '
ACCEPT *, A, B, Y1
```

```
10  Y2 = 0.5 * (A/Y1 + Y1)
    DELTA = ABS(Y2 - Y1)
    Y1 = Y2
    IF(B.GT.DELTA) GO TO 10
    TYPE *, 'Y2=', Y2
    STOP
    END
```

К итерационным циклам относится также вычисление сходящихся рядов с заданной степенью точности. Накопление суммы членов ряда производится до выполнения условия $|(n)| \leq \varepsilon$

(или $\left|\frac{(n)}{S_n}\right| \leq \varepsilon$), где (n) - величина очередного члена ряда; S_n - накопленная сумма.

Если все члены ряда положительны, то модуль их выделять не обязательно.

Пример. Вычислить с заданной относительной степенью точности

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

сумму членов ряда S области значений $0 < X \leq 1$

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

Примечание. 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

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

которое удобно использовать при составлении следующей программы:

```
    REAL X, E, R, B, S
    TYPE *, 'введите значение X и E'
    ACCEPT *, X, E
    R = 1.
    B = 1.
    S = 0.
```

```
1Ø    R=R*X/B
      S=S+R
      B=B+1.
      IF(E.GT.(R/S))GO TO 1Ø
      TYPE *,' сумма членов ряда=',S
      STOP
      END
```

Для случая знакопеременного ряда в рассмотренную программу нужно ввести следующую конструкцию:

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

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

```
      D=1              (или D = -I)
       . . .
      E=-D
```

Получаемое в каждом цикле значение D будет знакопеременной единицей.

5.3. Оператор цикла DO

Из предыдущих примеров ясно, что для организации цикла надо указать:

первый оператор в группе повторяющихся операторов (начало цикла);

последний оператор в этой группе (конец цикла);

сколько раз выполнить эту группу.

Оператор DO реализует эти три условия и дает некоторые дополнительные возможности.

Общий вид оператора цикла

$DO\ m\ I = n_1, n_2, n_3$,

где m – метка последнего оператора повторяющейся группы:

I – параметр цикла (простая целая переменная), является счетчиком цикла;

n_1, n_2, n_3 – целые числа или целые переменные и выражения, которые должны быть определены к моменту выполнения оператора *DO*.

Местоположение оператора *DO* определяет начало цикла, оператор с меткой m – конец цикла. Чаще всего в качестве последнего оператора цикла используют оператор *CONTINUE*. Это "пустой" оператор, который не указывает на выполнение каких-либо команд, но может иметь метку и таким образом указывать место конца цикла. Рекомендуется использовать этот оператор или оператор присваивания.

При входе в цикл параметр "I" получает значение n_1, затем работают операторы цикла до метки m. Метка m возвращает на начало цикла *DO*, переменная I получает новое значение, увеличенное на n_3 (шаг), т.е. во втором проходе цикла $I = n_1 + n_3$, в третьем $I = n_1 + 2n_3$ и т.д.

Выполнение цикла заканчивается при $I > n_2$, т.е. происходит выход из цикла.

Таким образом n_1 – начальное значение "I", n_2 – конечное значение "I", n_3 – шаг.

Если приращение (шаг) цикла $n_3 = 1$, то оператор можно записать короче: *DO* m $I = n_1, n_2$.

Рассмотрим примеры построения циклических программ.

П р о с т о й ц и к л

Пример 1. Составить программу вычисления конечного 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
S = S + 1./N
2 CONTINUE
```

Примечание. Если в записи *1./N* опустить точку, т.е. *1/N*, то при делении целого числа "I" на целую переменную *N* в результате для всех $N > 1$ получится "0", т.е. неверный результат.

Пример 2. Составить программу табулирования функции $F(x) = x + \sin x$ для n значений аргумента, изменяющегося в интервале $XN \leq X \leq XK$ с шагом H.

Фрагмент программы будет иметь вид

```
N=(XK-XN)/H+1          (расчет количества циклов)
DO 1 I=1,N
X=XN+(I-1)*H
F=X+SIN(X)
TYPE *,'X=',X,'F=',F
1 CONTINUE
```

Циклы с массивами

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

```
      REAL A(200)
      DO 3 K=2,200,2
      A(K-1)=A(K-1)**2
      A(K) =A(K)**3
3     CONTINUE
```

В некоторых программах требуется использовать одновременно прямой и обратный счетчики. В этих случаях приходится строить вспомогательные конструкции.

Пример 4. Составить программу переписи массива М1 (20) в обратном порядке:

```
      REAL M1(20),M2(20)
      ...
      DO 2 I=1,20
      K=21-I  (обратный счетчик)
      M2(K)=M1(I)
2     CONTINUE
```

В данной программе инвертированный массив будет уже иметь другое наименование - М2, и в памяти машины образуется два массива: один - с прямым расположением элементов (М1), другой - с обратным (М2).

Если же нужно произвести перемещение элементов внутри одного и того же массива М1, то можно дополнить приведенную программу еще

одним циклом переписи массива M2 (20) в M1 (20). Однако более интересным представляется следующий вариант программы:

```
      REAL   M1(2Ф)
      . . .
      DO 2  I= 1,1Ф
      K  = 21-I
      R  = M1(I)      ┐
      M1(I) = M1(K)   ├ пересылка
      M1(K) = R       ┘
2     CONTINUE
```

Здесь применена схема перемещения элементов массива с использованием так называемого "Посредника" (рис. 5).

Р и с. 5

В качестве "посредника" берется любая свободная ячейка (в примере с именем R), которая служит как бы промежуточной памятью для сохранения элементов массива.

Внимание! Количество циклов должно быть вдвое меньше количества элементов в массиве.

Пример 5. Вычисление полиномов. 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:

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

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 массиве нулевые индексы употреблять нельзя, сдвигаем индексы при коэффициентах a_i на единицу и принимаем начальное значение $P = a_6$.

Фрагмент программы имеет вид

```
      REAL A(6)
      ...
      P=A(6)
      DO 3 I=5,1,-1
      P=P*X+A(I)
3*    CONTINUE
```

Примечание. Обратите внимание, в операторе *DO* использован шаг $h_3 = -1$. Это допустимо для языка ФОРТРАН СМ ЭВМ.

В л о ж е н н ы е ц и к л ы

Цикл *DO* может содержать внутри один или несколько других циклов *DO* . При этом не допускается "перекрытий" циклов: включаемый цикл – в л о ж е н н ы й – должен полностью находиться внутри включающего внешнего.

Пример 6. Определить сумму элементов матрицы B размером 10 x 6:

```
      REAL B(10,6)
      ...
      S=0
      DO 1 I=1,10
      DO 2 K=1,6
      S=S+B(I,K)
2     CONTINUE
1     CONTINUE
```

При работе с вложенными циклами следует внимательно использовать передачи управления, учитывая, что вход в цикл возможен только через его начало, а выход из любого места.

На рис. 6,а схематически изображены допустимые, а на рис.6,б – недопустимые передачи управления для вложенных циклов.

Пример 7. Задача сортировки. Составить программу расстановки элементов массива $X(N)$ в порядке возрастания для N = 100 элементов.

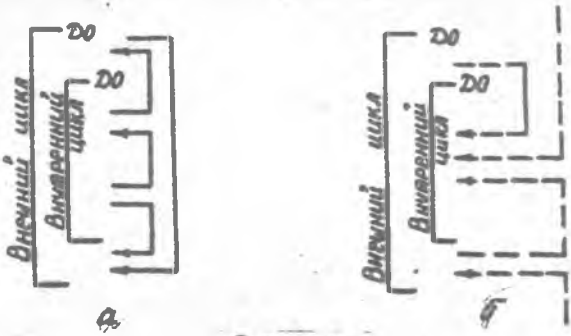


Рис. 6

Алгоритм программы таков, что в массиве X сначала находим минимальный элемент и переставляем его на первое место. Затем находим, начиная со второго элемента, минимальный элемент в массиве X. и переставляем его на второе место и т.д.

Во внутреннем цикле решается задача нахождения минимального элемента, а во внешнем - пересылка его на нужное место. Соответственно, первый элемент отсылается на место, которое занимал минимальный элемент. Заметим, что понадобятся дополнительные ячейки: XMN - для хранения минимального значения, M - для хранения номера (индекса) минимального элемента. Дополнительной резервной ячейки для "посредника" при пересылке не потребуется, т.к. значение минимального элемента имеется в двух ячейках: XMN и X(M).

Программа имеет вид

```
      REAL X(100), XMN
      INTEGER I, K, M, N
      . . .
      DO 2 I = 1,99
      K = I+1
      XMN=X(I)
      M=I
      DO 3 N=K,100
      IF(X(N).GE.XMN) GO TO 3
      XMN=X(N)
      M=N
    3 CONTINUE
      X(M)=X(I)
      X(I)=XMN
    2 CONTINUE
```