FORTRAN 文法の基礎(初級編)

2009-04-16 泉 聡志

１、はじめに

FORTRAN は数あるプログラム言語の中で最も数値計算に適した言語であり、かつ最もかんたんである。加えて、FORTRAN を使って数値計算プログラムを作成する工学者は、最小限のことを知っていれば良く、高度な知識は要求されない。

また、多くのプログラミングは、scratch から作らず、ベースとなるものを真似て改造して使う場合が多い。プログラミングにこうあるべきという型はないので、人のうまいところを取り入れながら、自分が組みやすいスタイルを確立していけばいいと考えられる。

ここでの文法は、基本的に fortran90 をベースとするが、fortran77 でもコンパイルできるレベルとする。g77 などの fortran77 コンパイラは fortran90 との表記の違いに対応しているので問題なくコンパイルできるはずである。

２、実行するには？

FORTRAN プログラムは単なるテキスト(文)であり、これを実行するためにはコンパイルという作業をして、実行形式のバイナリ(2 進法の機械語)を作成する必要がある。

%f77 test.f

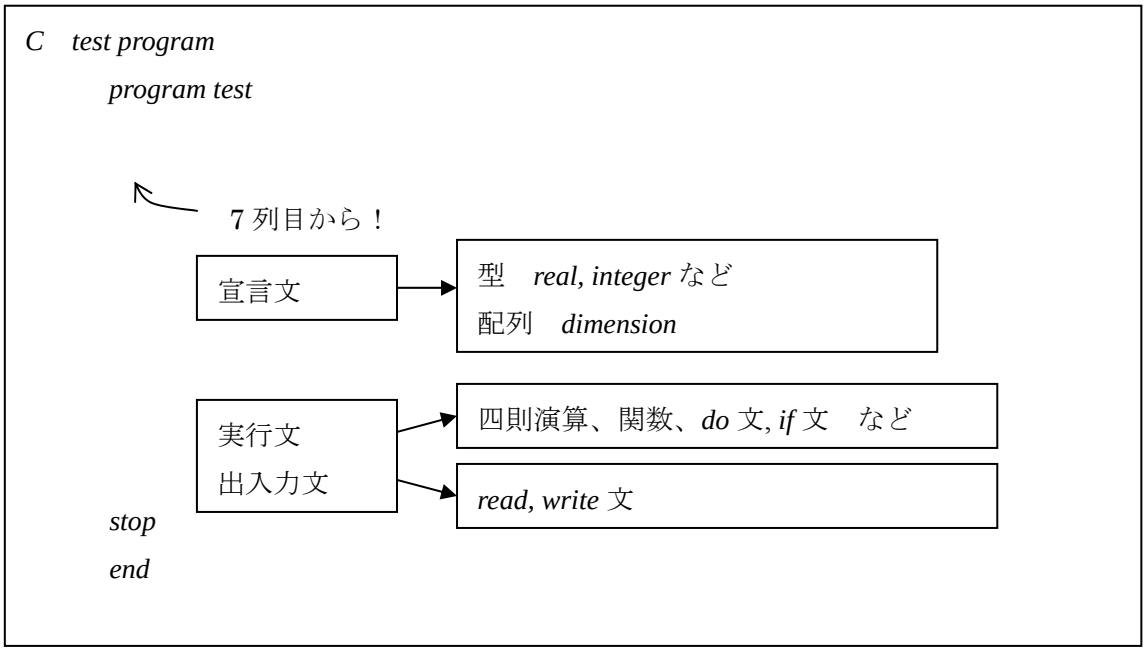
とすると、a.out というファイルが作られるので、

%a.out

とすれば実行できる。

３、FORTRAN形式の暗黙の了解

- FORTRAN は、*program test*　から始まり、*stop* と *end* で終わる。
- 注釈（コメント）は一列目に‘*C*’をいれる。もしくは、文末に’!’を入れて、その後に書く。
- 宣言文、実行文＆出入力文という順番で書く。ただし、変数の宣言をしなくとも良い場合がある。
- 小文字でも大文字でも区別はない。
- 一行に複数行の文を書く場合は’;’を入れる。　（例）stop; end



４、宣言文

４－１　変数の型

FORTRAN では、整数型と実数型を区別しなければならない。しかし、暗黙の了解では a～h と o～z は実数、i～n は整数となっているため、その規則に従えば、特に宣言しなくとも良い。それがどうしても嫌な人は *integer a1,a2,a3* や、*real n1,n2,n3* のように型を宣言する。数値計算をする場合、計算精度が要求されるため、一般に倍精度(単精度は 32bit、倍精度は 64bit の領域を使う)を用いることが多い、その場合は *double precision b1,b2,b3* とする。

```
integer a1,a2,a3
real n1,n2,n3
double precision b1,b2,b3
```

４－２　DIMENSION文

４－１のように、a1,a2,a3 とすべての変数を定義するのは面倒なので、それらを A(1),A(2),A(3)という 1 次元配列で定義する。ここでは、配列の大きさは３なので配列の次元を *dimension A(3)*と宣言する。整数で定義したい場合は *integer A(3)*。また、A(1),B(1),C(1)と配列が増えたときは、2 次元配列を使うと便利である。この場合、A(1,1),A(2,1),A(3,1)と書き、配列の次元は *dimension A(3,3)*とする。

```
dimension A(3)
dimension A1(3,3)
```

５、実行文

５－１　四則演算、関数

代表的なものを載せるが詳細は参考書を見て欲しい。

5.0 を 5.0E0 や 5.0d0 と書かないと、小数点以下の値がゼロになると保証されず、5.00010213 などといった値になってしまうので注意。

定義	FORTRAN
＋、－	*+, -*
×、÷	**, /*
べき乗　2^3	*2**3*
sin, cos	*sin( ) , cos( )*
平方根　√	*sqrt( )*
絶対値|A|	*abs( )*
指数　e^x	*exp( )*

定義	FORTRAN
整数　5	*5*
実数　5.0	*5.0 , 5.0E0*
実数　5.0×10^6	*5.0E6,5.0E+6*
実数　5.0×10^{-6}	*5.0E-6*
実数　0.001	*1.0E-3*
実数　1000.0	*1.0E+3*
倍精度　5.0×10^6	*5.0D+6*

５－２　DO文

FORTRAN の文法上、***最も使う文法***である。処理を繰り返し行うために使用する。

do 文の名前[任意]: do 文番号　I=開始数,終了数 に始まって、*enddo* に終わる。

```
do1: do I=1,5
A(I)=I*I
enddo do1
```

→ I を 1 から 5 まで変えて enddo まで繰り返す。A(I)には I＊I の値、つまり A(1)＝1 、A(2)＝ 4 、A(3)＝ 9 が入る。do1 は do ループの名前である。

参考に Fortran 77 では右のような記述になる。

```
    do10 I=1,5
    A(I)=I*I
10 continue
```

５－３　IF文

FORTRAN の文法上、DO 文と並んで良く使われるのが IF 文である。“もし～が～だったら、～する”というように、論理的判断を行う。*if (論理文) then* に始まって、*endif* に終わる。　論理が誤の場合の実行がない場合は *if (論理文)＋論理が正の場合の実行文*とする。

```
a=1
b=2
if ( a .gt. b ) then
   s=a
else
   s=b
endif
```

→

if 文開始

論理文：もし、a が b より大きいなら、その時は

論理が正の場合：s=a

else：さもなければ、

論理が誤の場合：s=b

if 文終了

※a は b より小さいので、s は 2 となる。

```
if ( a .GT. b ) s=a
if ( a .LE. b ) s=b
```


上と同じ意味

論理文は、数の大小を問うものであり（a が b より大きいなら、など）、（変数:a）＋関係演算子＋（変数:b）で構成される。関係演算子には以下のものがある。

.LT.	＜	(Less Than)
.LE.	<=	(Less than or Equal to)
.EQ.	＝	(EQual to)
.NE.	≠	(Not Equal to)
.GT.	＞	(Greater Than)
.GE.	>=	(Greater than or Equal to)

５－４　exit, cycle コマンド（Fortran77 ではgoto文を使う）

do ループの途中で、do ループを抜けたいときは exit 文を、その do ループの最後まで行って、次の do の反復ステップに行きたい時は、cycle を使う。

```
do I=1,5
A(I)=I*I
if ( A(I) .GT. 6.0 ) exit
enddo
```

DO 文のプログラムを修正、A(I)が 6.0 を超えたら、ループから抜け出すため、A(1),A(2)しか計算されない。

６、出入力文

６－１ READ文

- キーボードから入力する場合

  read(5,)a,b* とすると、プログラム実行時に 1.1,2.3 などと a,b に対応する２つの値を入力できるようになる

- ファイルから入力する場合

  入力する内容が書きこまれたファイルを空け(open)、読み込み、閉める(close)。

```
open(1,file='exam.dat')
read(1,*)a,b
close(1)
```

６－２　WRITE文

READ 文と要領は全く同じ。

- ディスプレイに出力する場合

  write(6,)a,b*

- ファイルへ出力する場合

```
open(2,file='exam.dat')
write(2,*)a,b
close(2)
```

６－３　FORMAT

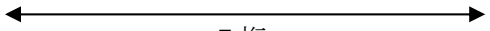
write(,*)*や *write(2,*)*の出力方式では、データがフリーフォーマットとなるため、見にくくなる。また、出力の桁数を指定しないとデータの精度の問題が生じる。よって、通常は、出力の書式をはっきりと設定する。

```
write(2,'(2e15.7)')a,b
```

FORMAT 文の方式は代表的なものを覚えておき、それ以外の書式は必要に応じて、参考書等を参照すれば良い。

2e15.7 の意味は、最初の 2 は入出力の変数の数、e は実数型のフォーマットの形式を表し、15 は変数を表示するのに必要なサイズ、7 は小数部分の桁数を表す。例えば、1.5432×10^{-7} を入出力する場合のフォーマットは以下のようになる。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		0	**.**	**1**	**5**	**4**	**3**	**2**	**0**	**0**	**E**	**－**	**0**	**6**

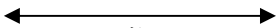


7 桁

データの保存に精度が要求される場合は、e17.9 や e21.13 として保存する桁数を多くする。

また、整数の場合の同様で、*2I4* なら変数が 2 つで、必要なサイズが 4 つとなる。154 を入出力する場合は以下のようになる。

1	2	3	4
	1	**5**	**4**



4 桁

あるフォーマットで出力されたファイルは、同じフォーマットで読みこまれなければいけないことに注意が必要。

７、練習問題

以上で、基本的なプログラムは組めるようになった。

練習問題として、以下の例題を与える。

A(50)は、3 の倍数が小さい順に並んだ配列である。

１）A(1)～A(50)までの和を求めよ。また、平均値を求めよ。

２）A の平方根が 7 以下のものをすべて抜き出し、その配列番号を表示せよ。

FORTRAN文法の基礎（中級編）　　2009-04-17 泉　聡志

１、動機付け

実際にプログラムを組むためには、この程度の知識では非効率的なことが多く、いくつかのテクニックが必要になる。ここでは、初級編と同じ順序でさらに高等なテクニックについて述べる。

２、実行するには

%f77 test.f –o test　実行ファイルを test と名前にする。

%f77 –c test.f

コンパイルのみをして実行ファイルは作らない。*test.o* というオブジェクトファイルが作られる。多数のプログラムを組み合わせてひとつの実行ファイルを作るときに、

%f77 main.f test.o –o test

とする。通常は makefile を作り、これらを自動的に行う。Makefile を一から作るのは難しいので大抵は誰かが作ったものを修正して使う。

３、宣言文

３－１　変数の型

i～n は整数、他は単精度の実数というきまりを、i～n は整数、他は倍精度の実数とするコマンド（プログラムの一番最初につける）

```
implicit integer (i-n)
implicit double precision (a-h,o-z)
```

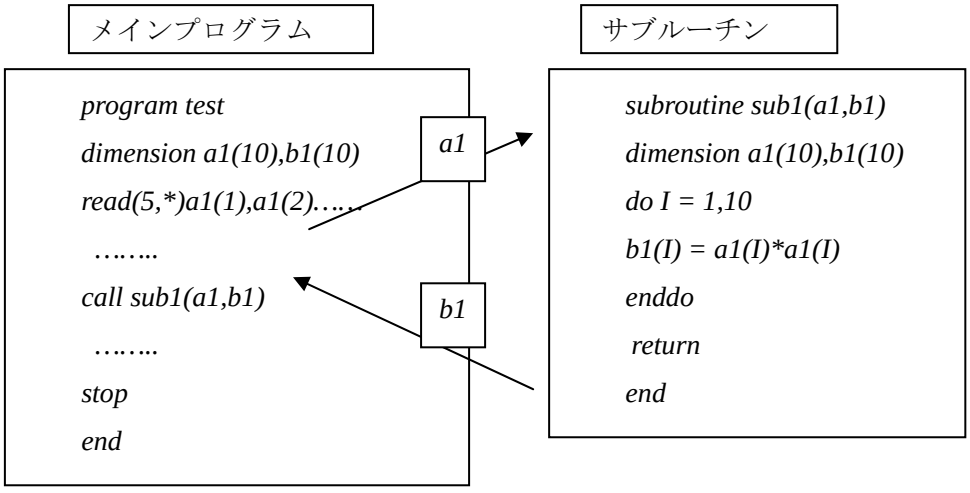
３－２　character

ファイル名などを管理する場合は、文字型を宣言しなければならない。

character filename*60 で 60 文字の filename という文字型の変数となる。

３－３　subroutine (サブルーチン)

サブルーチンの考え方はプログラム言語において、とても重要である。サブルーチンとは、メインプログラムより引数として受けた値に基づいて計算を行い(*a1*)、メインプログラムに戻す(*b1*)機能を持つ。初級編で紹介した DO 文や IF 文で出来たプログラムを複数回実行する場合や、ある特定の機能を持ったプログラムを自分のプログラムに組み入れる場合に便利である。



- サブルーチンの呼び出しは *call* 文で行う。サブルーチンに引き渡す変数、サブルーチンから引き渡される変数を引数として（ ）内に入れる。
- サブルーチンとメインプログラムの配列の次元・大きさは一致させること、名前は一致する必要はない（サブルーチンの a1,b1 の名前は変更可、ただし 10 は変更不可）。また、サブルーチンの配列宣言文は dimension a1(*),b1(*)と書いても良い。こうすることにより CALL するプログラムと整合する配列寸法を宣言したのと同じ意味になる。ただし 2 次元配列の場合 B(3,*)は許されても、B(*,*)や B(*,3)とすることは許されない。これにより、配列の寸法の変化に柔軟に対応できる。
- サブルーチンは *subroutine 名前 (引数)* で始まり、*return, end* で終わる。
- サブルーチンは何度呼び出してもかまわない。

３－４　パラメーター文

あらかじめ固定されている数値、または dimension にて用いられる数値は parameter 文内で定義することができて、配列の大きさを変えるとき便利である。整数以外の値を定義する場合は data 文が用いられる。

```
program test
parameter(N=10)
data A,B,C/1.0,2.0,3.0/
dimension a1(N),b1(N)
```

３－５　common文

通常は、サブルーチンの中の変数は local 変数と呼ばれ、メインプログラムや他のプログラムへは引数を通してしか影響しないが、common 文（共通ブロック）を使えば、プログラム全体で共通の変数が定義できる。つまり、プログラムのどこでも、その変数を変化させることができる。

メインプログラム

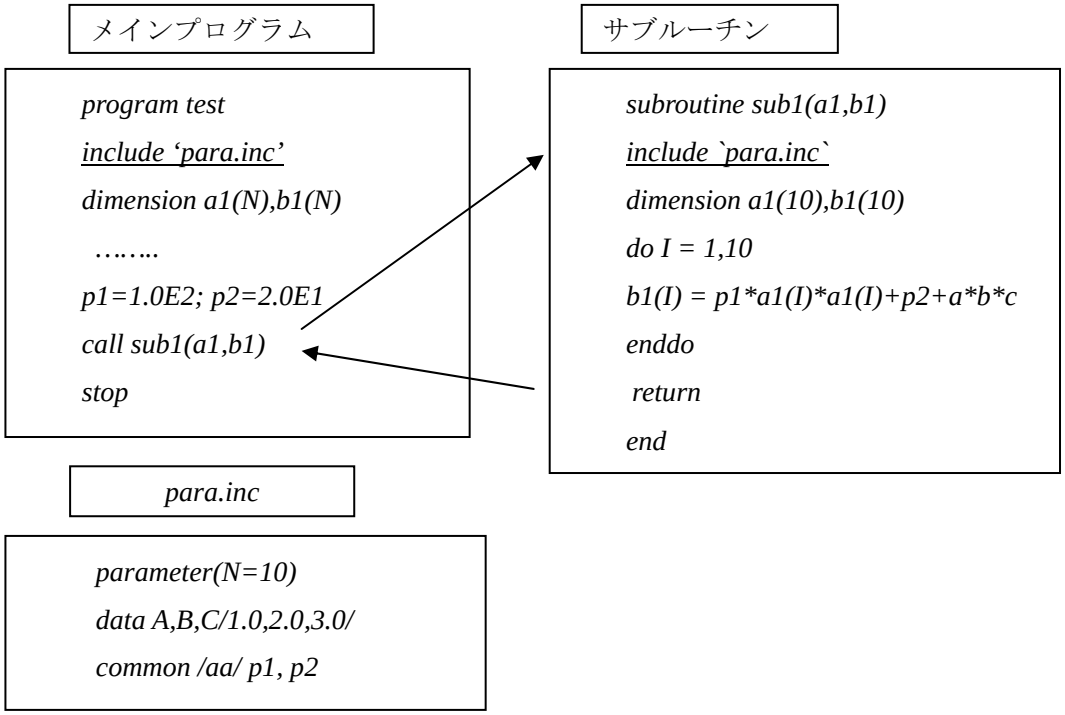
```
program test
common /aa/ p1, p2
dimension a1(10),b1(10)
 ……..
p1=1.0E2; p2=2.0E1
call sub1(a1,b1)
stop
```

サブルーチン

```
subroutine sub1(a1,b1)
common /aa/ p1,p2
dimension a1(10),b1(10)
do I = 1,10
b1(I) = p1*a1(I)*a1(I)+p2
enddo
 return
end
```

３－５　include文

あらかじめ固定されている変数や、共有される変数を include ファイルの中に書いておいて、すべてのサブルーチンの最初に、それを呼び出しておくと便利である。共通の値等が変わったときに、include ファイルのみの変更で対応でき、サブルーチンの修正がいらない。



４、実行文

４－１　配列の演算

例えば、Fortran77 と Fortran90 での行列の足し算と掛け算を比較する。

Fortran 77 式	Fortran 90 式
do I =1,3 *do J =1,3* *C(I, J) = A(I, J) + B(I, J)* *enddo* *enddo*	*C = A + B*

Fortran 77 式	Fortran 90 式
do I =1,3 *do J =1,3* *do k =1,3* *C(I, J) = A(I, k) * B(k, J)* *enddo; enddo; enddo*	*C = A * B*

逆行列や固有値は専用のサブルーチンが必要

４－２　IF文の使い方

a が b より大きくかつ、c が d より大きいという条件の IF 文

if ((a .gt. b) .and. (c .gt. d)) then

a が b より大きいもしくは、c が d より大きいという条件の IF 文

if ((a .gt. b) .or. (c .gt. d)) then

４－３　stop文

計算を終わらせてしまう場合 stop 文を用いる。

if (a .gt. b) stop　　条件を満たせば、その時点で計算が終了する。

プログラムにバグがある際に、stop 文を色々な位置に入れてみることにより、どこで止まっているかがわかる。

５．出入力文

５－１　READ、WRITE（配列対応）

read(1,'(10e15.7)')(A(I),I=1,10) → A(1)～A(10)を 1 行で読むことが出来る。

```
do J = 1,10
read(1, '(10e15.7)')(A(I,J),I=1,10)
enddo
```

→ 2 次元配列を 1 行で読みこむことはできない。

５－２　WRITE文（ユーザとの対話）

プログラムの途中や、データ入力を促す際に WRITE 文を使うと効率的である。

```
write(6,*)'please input data!'
read(5,*)a1,b1
write(6,*)'a1=',a1,'b1=',b1
```

‘ ‘で囲むとその部分が表示される。変数は“,”区切る必要がある。

まとめ

実際のソフト設計で一番重要なものはアルゴリズム（計算手法）の開発である。

代表的な数値計算アルゴリズムは、本や HP（LAPACK で）参照でき、必要に応じてサブルーチンを入手することも可能である。

[参考書]

[1]Fortran77 による数値計算ソフトウェア、渡部　力、丸善

[2]行列計算ソフトウェア、小国　力、丸善　　など

[HP]

[1] http://jjoo.hp.infoseek.co.jp/tips/fortlink/　Fortranリンク集

[2] http://www.netlib.org/lapack/　LAPACK -- Linear Algebra PACKage

演習１）３×３の行列の足し算と掛け算のサブルーチンを作成せよ。また、行列をファイルから読みこみ、足し算と掛け算を行うプログラムを作成せよ。

演習２）１次元配列（A(1),A(2),A(3),…….A(100)）をソーティング（小さいものから順にならべる）するサブルーチンを作成せよ。

演習３）大きさ N（任意）の 2 次元配列 A(N,N)とベクトル B(N),C(N)が **A＊B＝C** の関係にある。**A** と **C** が既知の時、**B** を求めるサブルーチンを作成せよ。（ガウスの消去法、参考書[1][2]を参照）

- **補足**（FUNCTION文）

  SUBROUTINE と異なり、関数として取り扱われ、出力変数を引数に必要としない。メインプログラムでは

  x=func1(a,b)

  というように sin や exp と同様の使い方をする(それらは標準関数)。

```
function func1(a,b)
func1 = a**a + b**b
return
end
```