Fortran
プログラミング入門

中久喜伴益
広島大学大学院理学研究科
地球惑星システム学専攻

プログラミング言語

機械語

CPUの命令をそのまま羅列したもの。２(16)進法

低級プログラミング言語

CPUの命令をそのまま英単語に近い単語で置き換えたもの

アセンブラ

高級プログラミング言語

数学や自然言語に近い形で処理を書いたもの

インタープリタ: 逐次機械語に翻訳しながら実行

Perl, BASICなど

コンパイラ: 機械語にまとめて変換して実行可能ファイル作成

Fortran, C, Pascalなど

プログラムの構造

入力

処理をさせるデータをコンピュータに渡す

入力元: キーボード，ファイル(ディスク)

処理(演算)

入力したデータを加工、データに基づいて計算

出力

加工したデータや計算結果を得る

出力先: 画面，プリンタ，ファイル(ディスク)

Fortran90

FORTRAN: FORmula TRAnsrator

数値計算に適したコンパイラ言語

特徴

数値計算の分野でよく使用されている

数式をほぼ数学での式と同様に書ける

形式が硬く決まっていて書き方に個人差が比較的少ない

Fortran90

FORTRAN77に配列の動的割り付け，モジュールなど，便利な機能を追加

Fortran90の実行まで

プログラムの作成

プログラムの形式や文法に注意

```
gedit hello.f90 &
```

コンパイル

機械語からなる実行形式ファイルにまとめて変換

```
f95 -o hello hello.f90
```

文法や形式にバグがあるとエラーになるのでデバッグ

実行

実行ファイル名をコマンドと思い、キー入力
“./”は「現在いるディレクトリの中にある」という意味

```
./hello
```

実行時エラーが出た場合はプログラムをデバッグ
必要に応じて変数の値を出力させ，想定値通りかチェック

Fortran90の形式

固定形式と自由形式

１行の文字数とコメント・継続行・行番号の位置の違い

どちらも大文字・小文字を区別していない

空白は無視

Fortran90の形式

固定形式: 拡張子“.f”

１行72文字

１文字目: コメント(cまたは!)，　２～５文字目: 行番号

６文字目: 継続記号

７～72文字目: 本文

自由形式: 拡張子“.f90”

１行132文字

コメントは!のみ

行番号は使わない (exitなどを使用)

本文はどこに書いても良い

次の行に継続させる場合は最後に&を書く

Fortran90の文法 (1): プログラム単位

program文

プログラム単位の始まり

```
program hello
```

サブルーチンプログラムの場合は，subroutine文

end program文

プログラム単位の終わり

```
end program hello
```

stop文

プログラム実行の中止

```
stop
```

Fortran90の文法 (2): 入出力

出力文

変数の値を外部装置へ出力する

```
write (6,*) A
write (10,*) 'hello'
```

( )の中の6や10は出力先装置番号

6は標準出力(ディスプレイ), *と書くことも出来る

10(5,6以外の数字)はファイル:

ファイル名はfort.10またはopen文で結びつけられた名前

*のところはフォーマット指定, *だとデフォルト

フォーマットを指定したいときは*のかわりにフォーマット指定の文字を値として持つ文字変数を入れる

```
fmt = 'F10.5,E12.5'
write (10,fmt) A, B
```

Fortran90の文法 (2): 入出力

入力文

変数に値を外部装置から入力する

```
read (5,*) A
read (10,*) B
```

( )の中の5や10は入力元装置番号

5は標準入力装置(キーボード)), *と書くことも出来る

10 (5,6以外の数字) はファイル:

ファイル名はfort.10またはopen文で結びつけられた名前

Fortran90の文法 (2): 入出力

補助入出力文

入出力装置番号とファイル名を結び付ける

```
open (10,file='outfile.dat')
open (11,file='infile.dat,status='old')
```

( )の中の10や11は装置番号

file='ファイル名'

でファイルを指定

statusはファイルの状態(既存か新規かなど)

status='new'

とすると新規ファイル

既存ファイルを指定しているとエラー(上書き防止)

Fortran90の文法 (3): 変数と定数

変数

変化する値を入れておく記号．数学の変数とほぼ同じ

255文字まで使える

```
A, b, Tmp, ...
```

値はプログラム実行時に変化する: 宣言文あるいは代入文

パラメータ(名前付き定数)

プログラム実行時に変化しない数値を記号で置き換えたもの

```
real(4), parameter:: SecY=365.2422*24.0*3600.0, ...
```

Fortran90の文法 (4): 変数の型

宣言文

変数の種類を指定(宣言)する

```
implicit none
```

暗黙の型宣言はしないことを宣言

変数は必ず宣言したものを使用する

```
real:: A, x, b
```

A, x, bが単精度(４バイト)実数であることを宣言

```
real(8):: Tmp, pres, vx, vz
```

Tmp, pres, vx, vzが倍精度(８バイト)実数であることを宣言

```
integer:: i, j, itr, ntime
```

i, j itr, ntimeが単精度(４バイト)整数であることを宣言

```
character(40):: fileA, data1
```

fileA, data1が40バイト(文字)の文字変数であることを宣言

Fortran90の文法 (5): 代入文

代入文

右辺の計算結果を左辺の変数に代入する

```
V=4.0d0/3.0d0*pi*r**3
```

左辺は必ず１つの変数

右辺は計算式や値，文字列など左辺の変数と同じ型のもの

数学演算

四則演算 (計算順序は数学と同じ)

```
+, -, *, /: A/b
```

べき乗

```
**: A**n
```

括弧 (小括弧のみを使用)

```
( )
```

Fortran90の文法 (6): 関数

関数

関数にはユーザーが定義できるユーザー定義関数と最初から組み込まれた組み込み関数がある

関数の呼び出し方: `x=float(i)`

( )の中の変数(またはパラメータ・式)を引数と呼ぶ

組み込み関数

数学で良く使用する数学関数のほか, 変数の型を変換する型変換関数がある

Fortran90の文法 (7): 組み込み関数

数学関数

三角関数: sin(x), cos(x), atan(x), tanh(x)...

指数／対数関数: exp(x), log(x), log10(x)

平方根: sqrt(x)

絶対値・符号変更: abs(x), sign(a,b)

剰余関数:mod(a,b), mod(a,b)=a-real(int(a/b)*int(b))

最大・最小値:max(a,b, ...), min(a,b, ...)

x やa, bは引数

引数と関数値は同じ型(引数が倍精度実数なら倍精度実数)

関数値の精度を名前で指定できる

　cos→docsと書けば倍精度で結果を出す

倍精度: dsin, dcos, dsqrt, dmod, dmax1...

Fortran90の文法 (7): 組み込み関数

数学関数

整数部(小数点以下切り捨て): `aint(a)`

整数部(小数点以下四捨五入）: `anint(a)`

型変換関数

整数→単精度実数: `real(i)` (代入先が単精度), `float(i)`

整数→倍精度実数: `real(i)` (代入先が倍精度), `dfloat(i)`

単精度→倍精度実数: `dble(a)`

倍精度→単精度実数: `sngl(d)`

実数→整数: `int(a)`, `int(d)`

i, a, dは引数

iは整数, aは単精度実数, dは倍精度実数

倍精度変換や整数変換の時に数値誤差に注意

Fortran90の文法 (8): 繰り返し処理

繰り返し: do文

do〜end do文を使用する

iが1からnまで繰り返す

```
do i = 1,n
  x = dx * dfloat(i)-dx*0.5d0
  y = A * sin(pi*x/L)
  write (6,*) x,y
end do
```

Fortran90の文法 (9): 配列

配列変数

行列・数列や数値計算の式で出てくる添え字付きの変数

a_iなどを`a(i)`のように括弧付きで書く

配列の宣言が必要

静的配列宣言

最初から配列に必要な数が分かっている場合

```
real(8):: a(100), b(0:100), x(n), y(-m,n), ...
real(8):: a(100,200), P(0:nx+1,0:ny+1), ...
```

宣言で( )の中は整数で数またはパラメータ

開始点を指定しないで(n)とすると(1:n)と同じ

Fortran90の文法 (9): 配列

動的配列宣言: Fortran90の新機能

配列の大きさを後で指定できる配列宣言

```
real(8),allocatable:: a(:), u(:,:,:), us(:,:),...
allocate (a(m))
allocate (u(1:nx+1,0:ny+1,0:nz+1), us(nx+1,ny),...)
```

配列の大きさの割り付け時に( )の中に変数を使用可能

動的割り付けが有効なのは手続き内のみ

注意：サブルーチンで定義した記憶領域はreturnで開放

Fortran90の文法 (9): 配列

整合配列

メインルーチンと配列の大きさを合わせる配列宣言

```
real(8):: u(1:nx+1,0:ny+1,0:nz+1), us(nx+1,ny),...
```

サブルーチン：配列の大きさに( )の中に変数を使用可能

注意：配列変数は引数

配列数の変数は引数あるいはグローバル変数

グローバル配列変数との使い分け

プログラムの読みやすさ分かりやすさ

Fortran90の文法 (9): 分岐

分岐処理: if文

条件式の真偽により処理を選ぶ

単純if文

処理が１行のみ

```
if (mod(N,2)==0) write (6,*) 'N is an even number'
if (N>2 .and. N<=5) write (6,*) 'N is 3, 4, or 5'
```

if (条件式) 実行文

条件式(論理式)の値が真(.true.)の時、実行文を実行

関係演算子

a<b:	a<b
a≥b:	a>=b
a=b:	a==b
a≠b:	a/=b

論理演算子

論理否定:	.not.(論理式)
論理積:	.and.
論理和:	.or.

実行順: 算術式＞関係式＞.not.＞.and.>.or.

Fortran90の文法 (9): 分岐

ブロックif文

処理が複数のとき

```
if (mod(N,3)==0) then
     ...
     ...
  else if (mod(N,3)==2) then
     ...
  else
     ...
     ...
end if
```

`else if`は無しでも複数個でも良い

`...`のところに実行文・１行でも複数行でも可

最後は`end if`で終わる

Fortran90の文法 (10): プログラムの分割

プログラムの分割

複雑な処理を行うプログラムを効率的に開発する
プログラム構成要素ごとに分けて部品化
組み合わせて大規模な1つのプログラムとする

プログラムの分割: プログラム単位

メインプログラム (メインルーチン)
サブルーチン副プログラム
関数副プログラム (外部関数, ユーザー定義関数)
モジュール副プログラム
各々を1つのファイルにしてディレクトリに入れておくと便利

Fortran90の文法 (10): プログラムの分割

コンパイル

```
f95 -o exec.out main.f sub1.f sub2.f fun1.f...
```

用いるプログラム部品のファイル名を羅列 (順不同)

モジュールを含むプログラムのコンパイル

モジュールがコンパイルされたファイルがあらかじめある必要がある

Makefileを使用すると便利

コンパイルする順番を指定

Fortran90の文法 (10): プログラムの分割

サブルーチン

サブルーチン名そのものは値を持たない

引数で処理に関係する変数を受け渡しする

呼び出し側

```
call sub_Tmp(Tmp, k, dz, dt, ...)
```

call文によりサブルーチンが呼び出される

sub_Tmpは適当な名前

変数は呼び出しプログラムとサブルーチンで局所的に定義される(名前が同じでも引数に書かないと値を受け渡さない)

引数はサブルーチン側と同じ型

Fortran90の文法 (10): プログラムの分割

サブルーチン

サブルーチン名そのものは値を持たない

引数で処理に関係する変数をやりとりする

サブルーチン側

```
subroutine sub_Tmp(Tmp, k, dz, dt, ...)
...
real(8):: Tmp(0:n), k, dz, dt
...
Tmp(j) = k*(Tmp(j-1)+Tmp(j+1)-...
...
end subroutine sub_Tmp
```

subroutine文で始まり、end subroutine文で終わる

引数は呼び出し側と同じ型

名前は呼び出し側と必ずしも同じでなくても良い

出力は引数の中にある変数として呼び出し側に返される

Fortran90の文法 (10): プログラムの分割

関数副プログラム

副プログラムが値を持っている

関数の値を変数に代入文で代入することが出来る

呼び出し側

```
real(8):: func2
...
y=func2(x1,x2,a,b,...)
```

関数名の型を宣言をする

func1は適当な関数名

関数の呼び出し: 組み込み関数と同様

変数は呼び出しプログラムとサブルーチンで局所的に定義される(名前が同じでも引数に書かないと値を受け渡さない)

Fortran90の文法 (10): プログラムの分割

関数副プログラム

副プログラムが値を持っている

関数の値を変数に代入文で代入することが出来る

関数副プログラム側

```
function func2(x1,x2,a,b,...)
...
real(8):: func2
...
func2=a*x1**2-b*x2..
...
end function func2
```

function文で始まり、end function文で終わる

function文の中の引数は呼び出し側と同じ型

関数名に対する型宣言が副プログラム内でも必要

関数の値は代入文func2=...で定義される

Fortran90の文法 (10): プログラムの分割

モジュール副プログラム

プログラムの共通する部分(多くは宣言文)を別のファイルにするためのしくみ

呼び出しプログラム側

```
use mod1
```

mod1**は適当なモジュール名**

モジュール中にある変数は値も共有：グローバル変数
(旧FORTRANのcommon文と似ている)
サブルーチンの時引数に書く必要がなくなる

```
use mod2, only:: A, Tmp,..
```

値を共有する変数が一部ときにはonly以下に共有する変数

Fortran90の文法 (10): プログラムの分割

モジュール副プログラム

プログラムの共通する部分(多くは宣言文)を別のファイルにするためのしくみ

モジュール側

```
module mod1
...
real(8):: Tmp(0:n), dt, dz
...
end module mod1
```

module文で始まり、end module文で終わる

Fortran90の文法 (11): その他の機能

文字列の演算

文字列の長さのカウントや文字列の連結など

```
moji3=moji1//moji2
```

構造型

複数の型の値を持つことが出来るデータ

ポインタ

変数のあるメモリ上のアドレスを表す変数