プログラミングの手順

- プログラムを書く
  （言語でコンピュータが作業する手順を記述）
  FORTRAN, C, C++, Java, Basic, Pascal, … （高級言語）

- コンパイル（翻訳）する　（高級言語→機械語）
  コンパイルするプログラム＝コンパイラー

- 実行 （実行形式→OS）

プログラムを書く

- コンピュータは命令を1つずつ指示された通りに実行する。(手続き型)
- 数値計算は、主に数値データを数値演算して、ある結果を得る、という作業
- ここでは、数値的に与えられた問題を解く手順(数値計算のアルゴリズム)を扱う
  微分方程式の数値解法を学び、それを応用することが目標

FORTRANプログラムの構成 (1)

プログラム単位:

- 主プログラム
  （programからend programまで）
- 外部副プログラム
  - 関数 (function)
  - サブルーチン (subroutine)
- モジュール（この演習では扱わない）

FORTRANプログラムの構成 (2)

例： 演習のサンプルプログラム

```
program logistic1        ← ここから
  implicit none
  integer,parameter:: nmax=100
  integer n
  real a,x

  a=1.6
  x=0.2
  do n=0,nmax
    write(*,'(i5,f10.5)') n,x
    x=(1.0-x)*x*a
  end do
end program logistic1        ← ここまでが主プログラム
```

FORTRANプログラムの構成 (3)

主プログラムや副プログラムは

変数宣言部 と プログラム本文

からなる。

```
program logistic1
  implicit none
  integer,parameter:: nmax=100
  integer n
  real a,x

  a=1.6
  x=0.2
  do n=0,nmax
    write(*,'(i5,f10.5)') n,x
    x=(1.0-x)*x*a
  end do
end program logistic1
```

変数宣言部

プログラム本体

変数 (1)

- プログラム中でデータを参照するために
  データに名前をつけたもの
- 変数にはデータの種類に応じて“型”がある。

- プログラム中で変数を利用するには:
  - 変数を宣言する（変数宣言部）
  - 値を代入する （プログラム本体）
  - 値を参照する （プログラム本体）

変数 (2)

- 例えば、ある国の人口を扱う変数を整数型でプログラム中で使いたい、と思って、人口を表す変数名をNと名付ける場合

INTEGER N　（Nを整数型で宣言）

N=1000000　（Nに値（ある国の人口）を代入）

FORTRANのデータの型

型	記述方法
整数型	integer
実数型	real
複素数型	complex
論理型	logical
文字型	character

Implicit none … 暗黙の型宣言をおこなわない

定数の表現方法

- 整数

  123, +543, -98765
- 単精度実数

  1.234, 1.234e0, 0.01, 1.0e-2, -1.23e0
- 倍精度実数

  1.23d0, -1.23d0, 0.01d0, 0.01d-2
- 複素数

  a+bi を表すには実数の組を括弧で括って(a,b) と表す

  (1.23 -5.67)
- 文字表現はアポストロフィ ‘ もしくは引用符 “ で 括る

  'ABC' , "ABC“
- 論理定数は真であれば .true. 偽であれば .false.

代入、演算

- 値を代入する

  a=b （変数aにbの値を代入する）

- 四則演算

  +（加） -（減） *（乗） /（除）

  a+b, a-b, a*b, a/b

  ()でくくることもできる　(a+b)*c

- 組み込み関数

  平方根 sqrt　三角関数 sin, cos, tan など

  a=sqrt(b), y=sin(x)

べき乗　　a**2　a**0.5　自然対数 log()、常用対数 log10()

配列

- 変数として、スカラー値だけでなく、ベクトル値などを扱いたい
- 要素がN個の配列を使うには dimension を使って変数宣言する

- 例）要素が3つの実数配列
  real,dimension(3): a
- 各要素を参照するには a(1), a(2) などのようにする。

- 多次元の配列（行列など）については
  real,dimension(3,2): a　…（3×2の行列）
- 参照については a(1,1), a(1,2) … のようにする。

制御構文 (1)

- 繰り返し　DO 文

```
do n=nstart, nend
        ....
end do
```

do と end do で間で挟まれた文を nstart から nend までnの値を1ずつ増やしながら繰り返して実行する。

```
do x=1.0, 1.5, 0.1
        ...
end do
```

という書き方もできる。この場合は x を 1.0 から 1.5 まで 0.1 ずつ値を増やしながら繰り返し実行する。

DO文を使ったプログラムの例

- １から１０までの整数の和をとる

```
program sum
  implicit none
  integer n
  integer s
  s=0
  do n=1,10
    s=s+n
  end do
end program
```

制御構文 (2)

- 条件判断　IF文

```
IF (  ) THEN
        ....
END IF
```

()内の論理式が真であれば IF END IF でかこまれた文を実行する。
論理式に使う演算子は以下のものなどが使える。

== 等しい
/= 等しくない
>= 右辺より左辺が大きいか等しい
<= 右辺より左辺が小さいか等しい
> 右辺より左辺が大きい
< 右辺より左辺が小さい

IF文を使ったプログラムの例

- １から１００までの３の倍数を出力する

```
program mod
  implicit none
  integer n
  do n=1,100
    if (mod(n,3)==0) then
      write(*,*) n
    endif
  end do
end program
```

入出力

- 外部のファイルや端末（標準入力、標準出力）からデータのやりとりを行う
- 読み込みは read を使う
- 書き込みは write を使う
- ファイルを開くには open を使う
- 開いたファイルは必ず閉じる close
- ファイルの区別はファイル識別番号を使う
- 標準入力・出力は番号ではなく * を使う

出力の書式(フォーマット)

- 何も考えたくない場合

  write(*,*) ….
- 整数の桁数を指定する

  Ix 桁数を*x*で指定(符号含む)

  5桁の場合 I5
- 実数の桁数を指定する

  F*m.n* 全体の桁数*m*と小数点以下桁数*n*

  によって指定(符号含む)

  全体10桁、小数点以下5桁の場合 F10.5
- 実数を指数形式で指定した桁数で表示する

  E*m.n* 全体の桁数*m*と仮数部の小数点以下桁数*n*

  によって指定(符号と指数部の桁数を含む)

  全体13桁、小数点以下5桁の場合 E13.5

FORTRANプログラムの構成 (2)

例： 演習のサンプルプログラム

```
program logistic1
  implicit none
  integer,parameter:: nmax=100
  integer n
  real a,x

  a=1.6
  x=0.2
  do n=0,nmax
    write(*,'(i5,f10.5)') n,x
    x=(1.0-x)*x*a
  end do
end program logistic1
```

5桁の整数と全体10桁小数点
以下5桁の実数を表示する

コンピュータ内部の数値表現

- 浮動小数点形式

$$\bar{x} = \pm\left(d_1\beta^{-1} + d_2\beta^{-2} + d_3\beta^{-3} + \cdots + d_m\beta^{-m}\right) \times \beta^n$$

d_iは$0,\ldots,\beta-1$の整数，ただし，$d_1 \neq 0$

()の部分を仮数，βを基数，nを指数とよぶ

最近のコンピュータは，単精度実数が32ビット
そのうち1ビットが符号，23ビットが仮数，8ビットが指数
基数は2（2進法）をとる
（2進数の場合かならずd_1=1となるので仮数は実質24ビット）

表現できる値は2^{-128}～2^{127}（2.9×10^{-39}～1.7×10^{38}）

丸め誤差

- 任意の実数を有限桁数の浮動小数点で表現するので，表現誤差が含まれる（丸め誤差）

実数 x の丸め誤差は相対誤差として

$$\bar{x} = \pm\left(d_1\beta^{-1} + d_2\beta^{-2} + d_3\beta^{-3} + \cdots + d_m\beta^{-m}\right) \times \beta^n \quad \text{から}$$

$$\left|\frac{\bar{x} - x}{x}\right| < c\beta^{1-n}$$

$d_1=d_2=\ldots=d_m=1$ の時に誤差が最小　2^{-25}〜3×10^{-8}

$d_1=1, d_2=\ldots=d_m=0$ の時に誤差が最小　2^{-24}〜6×10^{-8}

有効桁数はおよそ6〜7桁

桁落ち

- 値の非常に近い同符号の2つの実数を引き算すると有効桁数が大幅に減少する

計算上の工夫ができる場合は、できるだけ回避する。

例： 2次方程式の解

$ax^2 + bx + c = 0$ の解 $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

$b^2 >> 4ac$ の場合は一方の解の分子で桁落ちが発生する

根と係数の関係式 $x_1 x_2 = \dfrac{c}{a}$ を使って回避する