情報処理概論
(Basic Theory of Information Processing)

第１４回：Fortran

概要

- Fortanの変数
- Fortanの制御構造
- Fortanのプロシジャー・関数

17　Fortran

- **Formular Tranlator** の略
- はじめての高級言語
- コンパイラー
- 科学技術計算
- Lapack などの様々な科学技術計算のプログラムは Fortran で書かれている。
- 本学科ならば，神田研・高橋研がシミュレーションに使っている。
- 手続き型のパラダイムに属する。
- FORTRAN, FORTRAN66, FORTRAN77, FORTRAN90, FORTRAN95

17.1　書き方

- 使用できる文字
  - 英字：A∼Z (大文字，小文字は区別されない)
  - 数字：0∼9
  - 記号：　, _, =, +, −, *, /, (,), ,, ., ’, :, !, ", %, &, ;, <, >, ?, $

- 固定形式
  - 1行あたりのカラム数は，72と決まっている(横に72文字)。
  - 桁(カラム)ごとに意味が決まっている。

カラム	内容
1	Cまたは*ならばその行はコメント
1 ～ 5	文番号(ジャンプ先などを指定するため)
6	前の行に続いていることを示すときに，0と空白以外の文字を記入
7 ～ 72	文の内容を書く

- 自由形式
  - Fortran 90で採用された。
  - 1行あたり132カラム
  - ;で区切れば複数の文を1行に書くことができる。

内容	書き方
コメント	!から文末までがコメントになる
文番号	文の前にスペースで区切って文番号を書く。
行の継続	最後に&をつけると，次行が継続行になる。 文字列の継続のためには，次行の先頭にも&をつける。

17.2 Hello World

hello.f

```
C
C     hello.f
C
      PROGRAM hello
C
      WRITE (*,*) 'Hello World'
C
      STOP
C
      END
```

- プログラムは，PRGRAM文で始まって，END文で終わる。
- WRITE文で出力する。第1引数で出力場所，第2引数でフォーマットを指定する。
- STOP文でプログラムの実行を終了させる。

17.3　データ型と宣言文

17.3.1　データ型

- 特徴：**複素数**が扱える。

データ型	キーワード	メモリ領域 (byte)	Javaで対応するもの
論理型	LOGICAL	4	boolean
整数型	INTEGER	2	short
整数型	INTEGER*4	4	int
単精度実数型	REAL*4	4	float
倍精度実数型	REAL*8	8	double
倍々精度実数型	REAL*16	16	
単精度複素数型	COMPLEX*4	4	
倍精度複素数型	COMPLEX*8	8	
文字型	CHARACTER*n	n	

17.3.2　変数名

- 使用可能な文字：英数字，_
- 文字数：6文字，80文字
  　バージョンによって異なる。また，長い変数名が使うことができても，最初の6文字で区別するというもののものあった。

- 変数を宣言しなくても変数として使うことができる (**暗黙の変数宣言**)。
- そのため，整数に関する変数名は，I, J, K, L, M, Nからはじめることが多い,
  - この規則を守らなくても，変数を宣言すればエラーとなるわけではない。
  - ただし，従った方が皆が読みやすい。
  - 現在は，暗黙の宣言を禁止して，変数を宣言して使うことが推奨されている。
- その他にも，型に応じて始める文字に関する次のような慣習がある。
  - C：文字型
  - R：実数型
  - Z：複素数型
  - L：論理型
- 異なる型の変数による算術演算はFortranではできない。
- 異なる整数型変数から整数型変数，実数型変数から実数型変数，複素数型変数から複素数型変数への代入では自動的に型変換が行われる。
  (Fortranでは，代入は算術演算には含まれない。)
  (バージョンや実装によって，異なる型の代入で自動的に変換されることもある。)
- それ以外の変換は，**関数**を使う。
- 変数の宣言方法の実例は，演算といっしょに示す。

17.3.3 演算

- 算術演算：+, -, *, /, ** (べき乗)
- 文字演算：// (文字列を連結する。)
- 関係演算子：

演算子	意味	例
.EQ.	equal to	A .EQ. Bは，AとBが等しいときに.TRUE.
.NE.	not equal to	A .NE. Bは，AとBが等しくないときに.TRUE.
.LT.	less than	A .LT. Bは，AがBより小さいときに.TRUE.
.LE.	less than or equal to	A .LE. Bは，AがB以下のときに.TRUE.
.GT.	greater than	A .GT. Bは，AがBより大きいときに.TRUE.
.GE.	greater tahn or equal to	A .GE. Bは，AがB以上のときに.TRUE.

- 論理演算子：

演算子	意味	例
.NOT.	negation	.NOT. Aは，Aが.FALSE.のときに.TRUE.
.AND.	and	A .AND. Bは，AとBが.TRUE.のときに.TRUE.
.OR.	or	A .OR. Bは，AまたはBが.TRUE.のときに.TRUE.
.EQV.	equal to	A .EQV. Bは，AとBの論理値が等しいときに.TRUE.
.NEQV.	not equal to	A .NEQV. Bは，AとBの論理値が異なるときに.TRUE.

- 演算子の優先順位：

** > *, / > +, - > // > 関係演算子 > .NOT. > .AND. > .OR. > EQV, NEQV > =

17.4　制御構造

- ラベル
- GOTO文
- IF文
- DO文 (Javaのfor文)

17.4.1　ラベル

- 1 ～ 5カラムに記す数字で，文にラベル付けする。
- 1つのプログラムの中で，同じラベルを使ってはいけない。
- GOTO文で，プログラムの実行を指定したラベルが付いた文に移すことができる。
- DO文の範囲(繰り返し実行する範囲)を指定する。

```
100 Y = X



200 CONTINUE
```

- CONTINUEは，特に実行する内容がなく，ラベルを付けたいだけのときに使われる。
  - Javaの continue とは異なる。
  - ちなみに，Fortranには break 文はないが，ラベルとGOTO文で実現できる。

17.4.2 GOTO 文

- GOTO ラベルで，プログラム実行をそのラベルが付いた文へ移す。
- ラベルが付いた文は，GOTO 文の前にあっても後にあってもよい。

```
100 Y = X
    Z = D
    A = Y
    GOTO 100
```

- 上の例だと，無限ループになる。

17.4.3　IF 文

- IFだけの場合：

```
IF（条件）THEN

  条件が成立したときに実行する文の並び

END IF
```

- ELSEもある場合：

```
IF（条件）THEN

  条件が成立したときに実行する文の並び

ELSE

  条件が成立しなかったときに実行する文の並び

END IF
```

17.4.4 DO 文

- DO ラベル 変数=初期値,終了値,増分
- ラベルの文までを繰り返し実行する。
- 「,増分」を省略すると，増分は1になる。
- 例：

```
      PROGRAM OneToTen
      implicit none

      INTEGER I, ISUM
      ISUM = 0
      DO 10 I = 1,10
        ISUM = ISUM + I
   10 CONTINUE

      WRITE(*, *) ISUM

      STOP
      END PROGRAM OneToTen
```

- GOTO文とIF文で書くこともできる。

```
      PROGRAM OneToTen
      implicit none

      INTEGER I, ISUM
      ISUM = 0
      I = 1
   20 CONTINUE
      ISUM = ISUM + I
      I = I + 1
      IF (I .LE. 10) THEN
        GOTO 20
      END IF

      WRITE(*, *) ISUM

      STOP
      END PROGRAM OneToTen
```

17.5 配列

- Javaの配列と基本的には同じ
- 宣言：

```
INTEGER IA(4), IB(4,3)
REAL    C(50), X(100,20)
```

  として，変数の領域の確保を行う。
- このとき，IA(1), IA(2), IA(3), IA(4)が使える。また，
  IB(1,1), IB(2,1), IB(3,1), IB(4,1),
  IB(1,2), IB(2,2), IB(3,2), IB(4,2),
  IB(1,3), IB(2,3), IB(3,3), IB(4,3)
  が使える。
- 行列のベクトルの掛け算：

```
PROGRAM fortranMatrix
implicit none
REAL U(3), V(4)
REAL A(4, 3), SUM;
INTEGER I, J;
U(1) = 1
U(2) = 3
```

```
      U(3) = -1
      DO 10 I = 1, 4
        A(I, 1) = I
        A(I, 2) = I ** 2
        A(I, 3) = I ** 3
   10 CONTINUE
C     Matrix-Vector multiplication
      DO 20 I = 1, 4
        SUM = 0.0
        DO 30 J = 1, 3
          SUM = SUM + A(I, J) * U(J)
   30   CONTINUE
        V(I) = SUM;
   20 CONTINUE
C     Display the results
      DO 40 I = 1, 4
        WRITE(*, 100) I, V(I)
   40 CONTINUE
  100 FORMAT(2HV(, I3, 4H) = , F8.2)
      END program fortranMatrix
```

17.6　副プログラム

- 副プログラム：javaのメソッドのようなも。
- サブルーチン副プログラム
  - CALL文を使って呼び出す。
  - 戻り値がない。
  - FORTRANはサブルーチンが**参照渡し**であるため，引数の値を変えると，読み出しもとでも変わっている。
  - 定義：

    SUBROUTINE サブルーチン名(仮引数，仮引数，...)
    引数の型の宣言

    処理

    END SUBROUTINE

- 関数副プログラム
  - 戻り値がある。
  - 呼び出しは，式の中で通常どおり呼び出す。
  - sin(X)，cos(X)など，組み込み関数が使える。
  - 定義：

    戻り値の型 FUNCTION 関数名(仮引数，仮引数，...)
    引数の型の宣言

    処理

    関数名 = 結果処理結果

    END FUNCTION

- 目的：Pが素数かどうか調べるサブルーチン
- 配列PAにnP個の素数が入っている。
- Pをそれらの素数で割ってみて，それらすべて割りきれなければisPが.TRUE.になり，そうでなければisPが.FALSE.になる。

```
      SUBROUTINE IsPrime(isP, P, nP, PA)
      implicit none
      LOGICAL isP
      INTEGER P, nP, PA(1000), I, P1

      DO 10 I = 1, nP
        P1 = PA(I)
        IF (P - P/P1*P1 .EQ. 0) THEN
          isP = .FALSE.
          RETURN
        END IF
   10 CONTINUE
      isP = .TRUE.
      RETURN
      END
```

- サブルーチン IsPrime を使って，素数を 1000 個計算する。

```
      program Prime
      implicit none

      INTEGER primeArray(1000)
      INTEGER nPrime, primeCand, I
      LOGICAL isP
      primeCand = 2
      nPrime    = 0

   10 CONTINUE
        CALL IsPrime(isP, PrimeCand, nPrime, PrimeArray)
        IF (isP) THEN
          nPrime = nPrime + 1
          primeArray(nPrime) = PrimeCand
        END IF
        primeCand = primeCand + 1
      IF (nPrime .LT. 1000) GOTO 10
```

```
   DO 20 I = 1, 1000
20   WRITE(*,*) I, primeArray(I)

   STOP
   end program Prime
```

17.7　その他

- モジュール
  　ソースプログラムで，共通に利用する宣言などをモジュール化し，useを使ってそのモジュールを組み入れることができる。

- COMMON文　サブルーチン間などでデータを共用する。

17.8　今日の最後に

- FORTRANは科学技術ではまだまだ現役
- 神田研，高橋研では使っている。
- ラベルとGOTO文で制御構造が複雑なスパゲティプログラムにならないように注意