2019年度前期　金曜4限

情報処理演習

第4回

2019/05/17

平山　修久

名古屋大学減災連携研究センター
Disaster Mitigation Research Center, NAGOYA UNIVERSITY

NAGOYA UNIVERSITY

前回までの復習

Step 1　プログラムを書く

- エディタ（メモ帳）でソースコードを入力し，ファイル「***.f90**」（拡張子）で保存。

Step 2　プログラムをコンパイルする

- コマンドプロンプトで，「**gfortran プログラム名**」で，「**a.exe**」の実行ファイルを生成。

Step 3　プログラムを実行

- コマンドプロンプトで，「**a.exe**」と入力してEnter

単純論理式

- 論理定数（.TRUE.と.FALSE.）

式1　関係演算子　式2

演算子		意味
<	または .LT.	より小さい
>	または .GT.	より大きい
.EQ.	または ==	等しい
<=	または .LE.	以下
>=	または .GE.	以上
/=	または .NE.	等しくない

NAGOYA UNIVERSITY

複合論理式

- .NOT.（否定），.AND.（論理積），.OR.（論理和），.EQV.（等価），.NEQV.（排他的論理和）

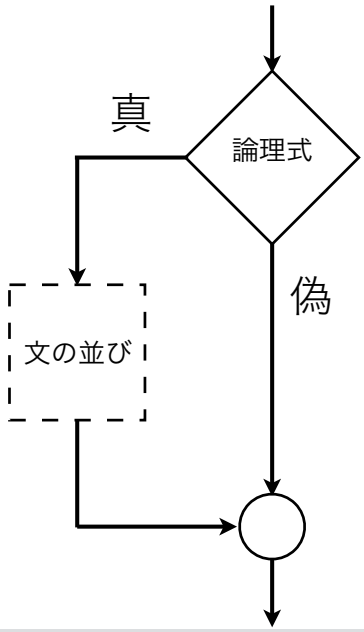
p	.NOT. p
.TRUE.	.FALSE.
.FALSE.	.TRUE.

p	q	p .AND. q	p .OR. q	p .EQV. q	p .NEQV. q
.TRUE.	.TRUE.	.TRUE.	.TRUE.	.FALSE.	.TRUE.
.TRUE.	.FALSE.	.FALSE.	.TRUE.	.FALSE.	.TRUE.
.FALSE.	.TRUE.	.FALSE.	.TRUE.	.FALSE.	.TURE.
.FALSE.	.FALSE.	.FALSE.	.FALSE.	.TRUE.	.FALSE.

IF構文

- 最も単純な選択構造

```
IF (論理式) THEN
    文の並び
END IF
```

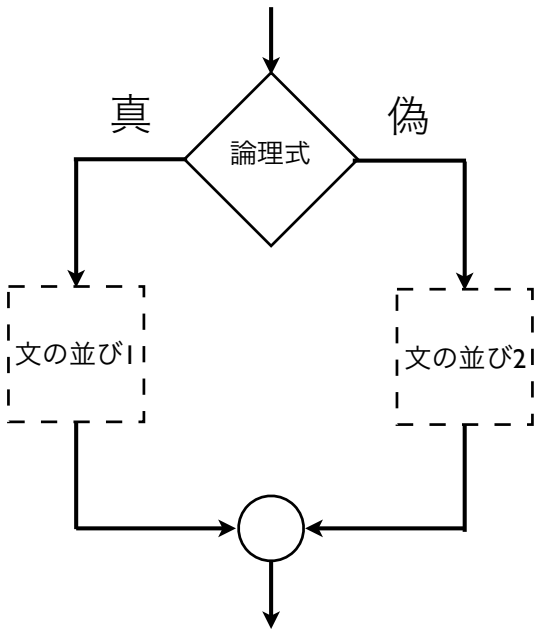



```
IF (X >= 0) THEN
    Y = X * X
    Z = SQRT(X)
END IF
IF (1.5 <= X .AND. x <= 2.5) PRINT *,X
```

NAGOYA UNIVERSITY

IF構文の汎用形式

```
IF (論理式) THEN
    文の並び1
ELSE
    文の並び2
END IF
```



IF-ELSE IF構文

- 3つ以上の選択肢が含まれるとき

```
IF (論理式1) THEN
    文の並び1
ELSE
    IF (論理式2) THEN
        文の並び2
    ELSE
        文の並び3
    END IF
END IF
```

- IF-ELSE IF構文を使うと

```
IF (論理式1) THEN
    文の並び1
ELSE IF (論理式2) THEN
    文の並び2
ELSE IF (論理式3) THEN
    文の並び3
ELSE
    文の並びn
END IF
```

NAGOYA UNIVERSITY

CASE構文

```
SELECT CASE (場合式)
    CASE (場合値リスト1)
        文の並び1
    CASE (場合値リスト2)
        文の並び2
    CASE (場合値リスト3)
        文の並び3
END SELECT
```

- 場合式は，整数式，文字式，論理式
- 場合値リストは，場合式が取りうる1つ以上の値を括弧で囲んだリストか，キーワードDEFAULT
- (値)【単一の値】，(値1:値2)【値1から値2までの範囲】，(値1:)【値1以上のすべての値の集合】，(:値2)【値2以下のすべての値の集合】

繰り返し実行：カウンタ制御DOループ

```
DO 制御変数 = 初期値, 限界値, 増分値
    文の並び
END DO
```

1.制御変数に初期値が代入される。

2.制御変数が限界値と比較され，次の条件を満たすかどうか調べる

- 増分値が正の場合は，限界値以下であるかどうか
- 増分値が負の場合は，限界値以上であるかどうか

3.条件を満たす場合は，ループ本体と呼ばれる文の並びが実行され，制御変数に増分値が加算されて，ステップ2が繰り返される。そうでない場合は繰り返しが終了する。

NAGOYA UNIVERSITY

繰り返し実行：カウンタ制御DOループ

```
DO Number = 1, 9
    PRINT *, Number, Number**2
END DO
```

```
MacPro2017:0517 nhirayama$ ./a.out
           1           1
           2           4
           3           9
           4          16
           5          25
           6          36
           7          49
           8          64
           9          81
```

- Numberは整数型変数，Numberが制御変数，初期値1，限界値9，増分値1
- 初期値，限界値，増分値には変数や式も指定できる。

```
READ *, Number
DO I = 1, Number
    Sum = Sum + I
END DO
```

1+2+3+ … + Number

繰り返し実行：汎用DOループ(1)

> Do-EXIT構文

```
DO
    文の並び1
    IF (論理式) EXIT
    文の並び2
END DO
```

1.文の並びに1または文の並び2は省略できる。

2.ループ本体を構成する文は，IF文の論理式が真になるまで繰り返される。IF文の論理式が真になると，その時点で繰り返しは終了し，END DOの次の文に実行が続く。

3.論理式が決して真にならない場合は，**無限ループ**となる。

NAGOYA UNIVERSITY

繰り返し実行：汎用DOループ(2)

> 事前テストループ

```
DO
    IF (論理式) EXIT
    文の並び
END DO
```

> 事後テストループ

```
DO
    文の並び
    IF (論理式) EXIT
END DO
```

繰り返し実行：汎用DOループ(3)

- Limitが与えられた時，1+…+NumberがLimitより大きくなる正の整数値Numberの最小値を求める。

```
Number = 0
Sum = 0
DO
    IF (Sum > Limit) EXIT
    Number = Number + 1
    Sum = Sum + Number
END DO
PRINT *, Number
```

NAGOYA UNIVERSITY

繰り返し実行：汎用DOループ(4)

- 問い合わせ制御入力ループ

```
DO
  PRINT *, “Enter temperature in degrees Celsius :”
  READ *, Celsius
  Fahrenheit = 9.0 / 5.0 * Celsius + 32.0
  PRINT *, Celsius, “(C) = “, Fahrenheit, “(F)”
  PRINT *, “More temperatures to convert(Y or N)?”
  READ *, Response
  IF (Response == “N”) EXIT
END DO
```

問い合わせ制御入力ループ

demo4_2

```
PROGRAM Demo4_2

        IMPLICIT NONE
        REAL :: Celsius, Fahrenheit
        CHARACTER(1) :: Response

        DO
                PRINT *, "Enter temperature in degrees Ce
                READ *, Celsius
                Fahrenheit = 9./5.*Celsius + 32.0
                PRINT *, Celsius, "degrees Celsius = ", F
                PRINT *
                PRINT *,"More temperatures to convert (Y
                READ *, Response
                IF (Response == "N") EXIT
        END DO

END PROGRAM Demo4_2
```

0517 — -bash — 128×72

```
MacPro2017:0517 nhirayama$ gfortran demo4_2.f90
MacPro2017:0517 nhirayama$ ./a.out
 Enter temperature in degrees Celsius:
0
   0.00000000     degrees Celsius =    32.0000000     degreed Fahrenheit.

 More temperatures to convert (Y or N)?
Y
 Enter temperature in degrees Celsius:
100
   100.000000     degrees Celsius =    212.000000     degreed Fahrenheit.

 More temperatures to convert (Y or N)?
u
 Enter temperature in degrees Celsius:
80
   80.0000000     degrees Celsius =    176.000000     degreed Fahrenheit.

 More temperatures to convert (Y or N)?
y
 Enter temperature in degrees Celsius:
-100
  -100.000000     degrees Celsius =   -148.000000     degreed Fahrenheit.

 More temperatures to convert (Y or N)?
n
 Enter temperature in degrees Celsius:
800
   800.000000     degrees Celsius =    1472.00000     degreed Fahrenheit.

 More temperatures to convert (Y or N)?
N
MacPro2017:0517 nhirayama$
```

NAGOYA UNIVERSITY

繰り返し実行：汎用DOループ(5)

- EXIT文は，END DO文の次の文に制御を移すことによってループの繰り返しを終了させる。
- 現在の繰り返しだけを途中で終了させる：**Do-CYCLE構文**

```
DO
    READ *, Celsius
    IF (Celsius < 0.0) THEN
        PRINT *, “/// Temperature must be 0 or above”
        CYCLE
    END IF
    文の並び
END DO
```

1.Celsiusに負の値が入力されると，メッセージが表示され，残りのループ文をスキップ。ループの繰り返しが新たに開始される。

課題4_1　フィボナッチ数列

- フィボナッチ数列 Fn = Fn-1 + Fn-2, F0 = 0, F1 = 1を用いて，n番目（n > 2）のフィボナッチ数，ならびに黄金比（Fn / Fn-1）の値を求めるプログラムを作成する。
- アルゴリズム
  1) nを入力する。
  2) (1)～(3)を繰り返し（i = 2からn）計算する。
     (1) Fn = Fn0 + Fn1を計算
     (2) Fn0にFn1を代入
     (3) Fn1にFnを代入
  3) Golden_ratioをFn1/Fn0により計算する。
  4) FnとGolden_ratioを表示する。

NAGOYA UNIVERSITY

課題4_2　数値積分

- 台形公式を用いて定積分の近似値を求める。

$$s = \frac{1}{2}\Delta x \sum_{k=0}^{n-1}(y_k + y_{k+1}) = \left(\frac{1}{2}(y_0 + y_n) + y_1 + y_2 + \ldots + y_{n-1}\right)\Delta x$$

- アルゴリズム
  1) nを入力する。
  2) 分割幅dxを計算する。
  3) 面積sを初期化する。
  4) (1)～(3)を繰り返し（i = 0からn）計算する。
     (1) x軸上のi番目の値xを計算する。
     (2) xに対応するyを求める。
     (3) 両端の点のとき，yを1/2倍して面積sに加算，その他はyをそのまま面積sに加算
  5) 面積sにdxを乗じて近似値を計算する。
  6) 面積sを表示する。

倍精度実数型変数の宣言

REAL：単精度実数（4バイト）

2^{-126}～2^{127}

```
a = 1.0e0
```

DOUBLE PRECISION：倍精度実数（8バイト）

2^{-1022}～2^{1023}

```
a = 1.0d0
```

DBLE(a)：倍精度実数への変換