

2021年度前期 情報処理演習

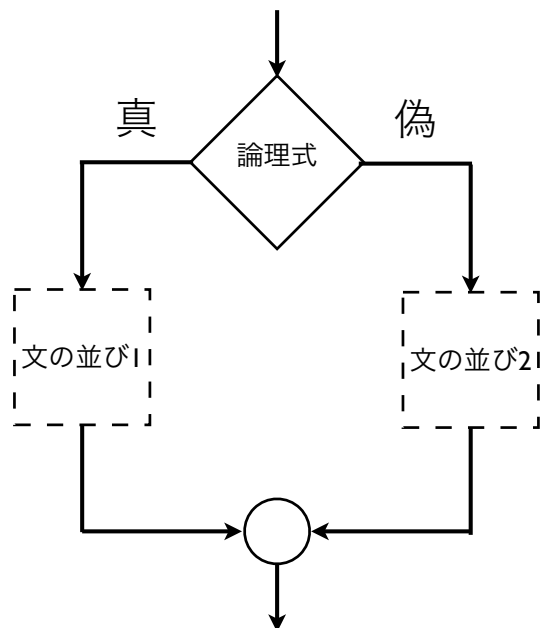
第5回
2021/05/19
平山 修久



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

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構文例とCASE構文例

```
IF (N >= 90) THEN
    PRINT *, "S"
ELSE IF (N >= 80) THEN
    PRINT *, "A"
ELSE IF (N >= 70) THEN
    PRINT *, "B"
ELSE IF (N >= 60) THEN
    PRINT *, "C"
ELSE
    PRINT *, "F"
END IF
```

```
SELECT CASE (N)
CASE (90:100)
    PRINT *, "S"
CASE (80:89)
    PRINT *, "A"
CASE (70:79)
    PRINT *, "B"
CASE (60:69)
    PRINT *, "C"
CASE DEFAULT
    PRINT *, "F"
END SELECT
```



Tips

- ー 条件分岐：研究においては、想定していない値となった場合について考えることが必要。0での除算や想定した入力データ以外の場合。
- ー CASE文の場合には、最後にDEFAULTを入れることが、思わぬ計算違いを防ぐ。場合式は**整数 (INTEGER)**

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



繰り返し



繰り返し実行：カウンタ制御DOループ

```
DO i = 10, 1, -2  
  文の並びi*i  
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。



繰り返し実行：カウンタ制御DOループ

MacPro2017:0517 nhirayama\$./a.out

1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81

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

- > 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。



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, $1+\dots+Number$ がLimitより大きくなる正の整数値Numberの最小値を求める。

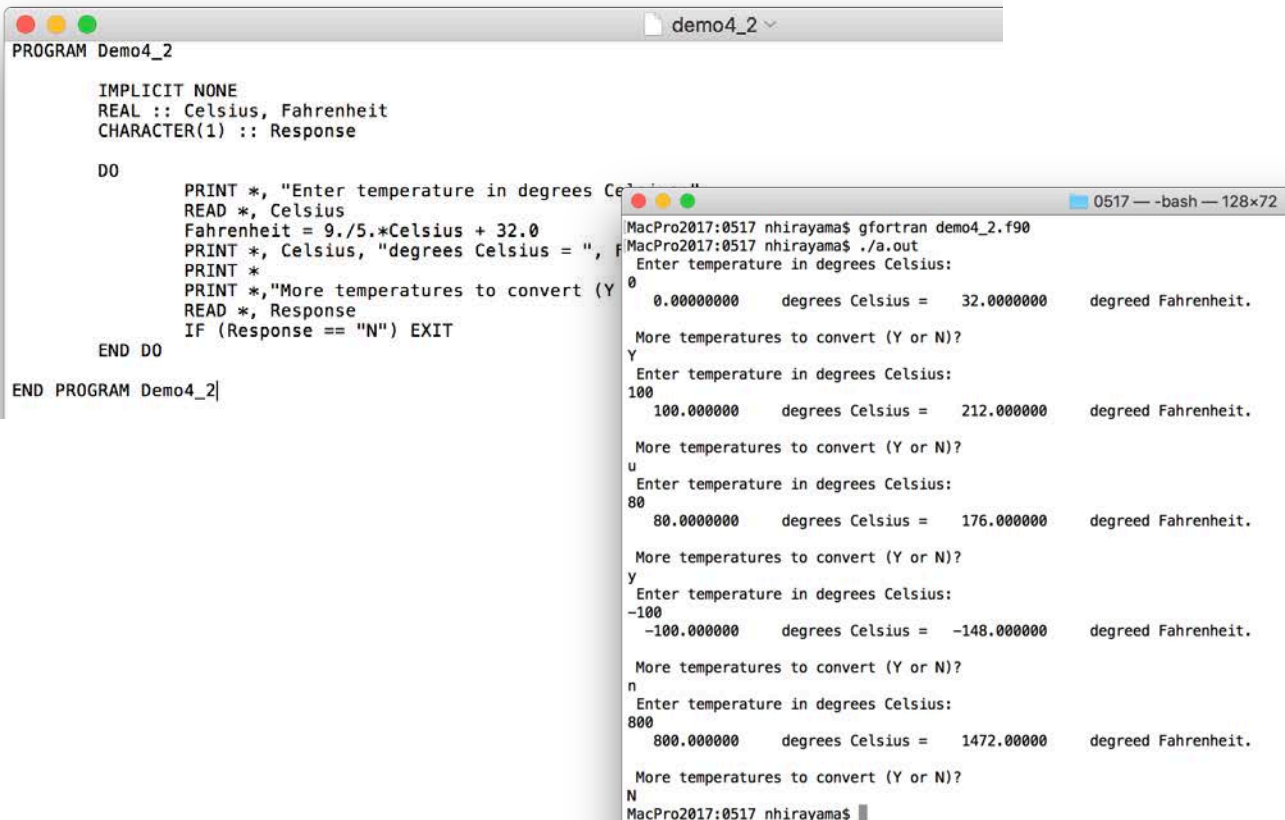
```
Number = 0
Sum = 0
DO
    IF (Sum > Limit) EXIT
    Number = Number + 1
    Sum = Sum + Number
END DO
PRINT *, Number
```

繰り返し実行：汎用DOループ(4)

＞ 問い合わせ制御入カループ

```
DO
  PRINT *, "Enter temperature in degrees Celsius : "
  READ *, Celsius
  Fahrenheit = 1.8 * Celsius + 32.0
  PRINT *, Celsius, "(C) = ", Fahrenheit, "(F)"
  PRINT *, "More temperatures to convert(Y or N)?"
  READ *, Response
  IF (Response == "N") EXIT
END DO
```

問い合わせ制御入カループ



```
PROGRAM Demo4_2
  IMPLICIT NONE
  REAL :: Celsius, Fahrenheit
  CHARACTER(1) :: Response

  DO
    PRINT *, "Enter temperature in degrees Celsius : "
    READ *, Celsius
    Fahrenheit = 1.8 * Celsius + 32.0
    PRINT *, Celsius, "degrees Celsius = ", Fahrenheit, "degree Fahrenheit."
    PRINT *, "More temperatures to convert (Y or N)?"
    READ *, Response
    IF (Response == "N") EXIT
  END DO
END PROGRAM Demo4_2
```

```
MacPro2017:0517 nhirayama$ gfortran demo4_2.f90
MacPro2017:0517 nhirayama$ ./a.out
Enter temperature in degrees Celsius:
0
0.0000000 degrees Celsius = 32.0000000 degree Fahrenheit.
More temperatures to convert (Y or N)?
Y
Enter temperature in degrees Celsius:
100
100.000000 degrees Celsius = 212.000000 degree Fahrenheit.
More temperatures to convert (Y or N)?
u
Enter temperature in degrees Celsius:
80
80.0000000 degrees Celsius = 176.000000 degree Fahrenheit.
More temperatures to convert (Y or N)?
y
Enter temperature in degrees Celsius:
-100
-100.000000 degrees Celsius = -148.000000 degree Fahrenheit.
More temperatures to convert (Y or N)?
n
Enter temperature in degrees Celsius:
800
800.000000 degrees Celsius = 1472.000000 degree Fahrenheit.
More temperatures to convert (Y or N)?
N
MacPro2017:0517 nhirayama$
```

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。

今日 (20210519) の課題 (1/2) フィボナッチ数列

1. フィボナッチ数列

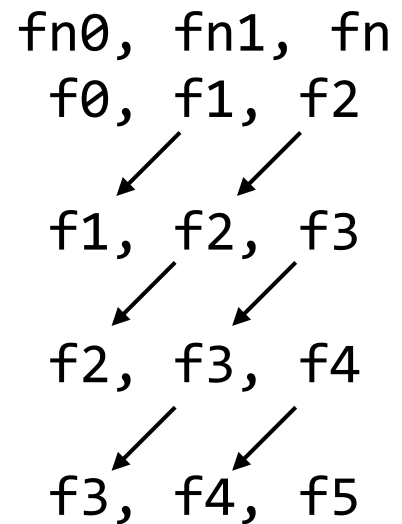
$$F_n = F_{n-1} + F_{n-2}, \quad F_0 = 0, \quad F_1 = 1$$

を用いて， n 番目 ($n > 2$) のフィボナッチ数，ならびに黄金比 F_n/F_{n-1} の値を求めるプログラムを作成する。

$n = 10$ ， $n = 25$ のときのフィボナッチ数，黄金比の値を計算する。

フィボナッチ数列-アルゴリズム

1. nを入力する。
2. 繰り返し計算 (i=2からn)
 - 1) $fn = fn0 + fn1$ を計算する。
 - 2) $fn0$ に $fn1$ を代入する。
 - 3) $fn1$ に fn を代入する。
3. 黄金比 $golden_ratio$ を計算する。
4. fn と $golden_ratio$ を表示する。



今日の課題 (20210519) (2/2)

数値積分

2. 台形公式

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

を用いて,

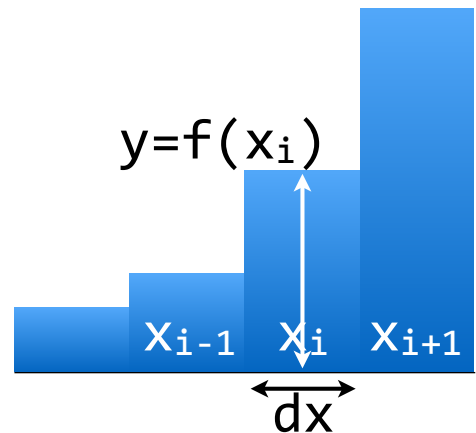
$$\int_0^1 x(x-1)(x-2) dx$$

の定積分の近似値を求めるプログラムを作成する。

$n = 100$, $n = 1,000$, $n = 10,000$ の時の定積分の近似解をそれぞれ求めよ。

数値積分－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. 繰り返し ($i=0$ から n) 計算
 - 1) x 軸上の i 番目の値 x を計算する。
 - 2) 値 x に対応する y を計算する。
 - 3) 両端の点のとき, $i=0$, $i=n$ のとき, y を0.5倍する。
 - 4) 面積 s に y を加算する。
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。



計算精度を上げるには

- 整数型: INTEGER ← 2バイト 16ビット
 $-2^{15} \sim 2^{15}-1 \rightarrow -32768 \sim 32767$
- 実数型: REAL(4) ← 単精度実数 (4バイト) 32ビット
 $0.117549435E-37 \sim 0.340282347E+39$ 精度7桁
 $a = 1.0$ もしくは $a = 1.0e0 \ 1.0e2$
DOUBLE PRECISION ← 倍精度実数 (8バイト)
REAL(8)
 $0.22250738585072014E-307 \sim$
 $0.17976931348623157E+309$ 精度16桁
 $a = 1.0d0$
 倍精度実数への変換 **DBLE(変数)**