情報処理　（第３回）　(2017/4/24 茂木)

本日のテーマ

- 変数
- 演算子・式
- read 文，write 文

1　解答例

Q.1 足し算

```
c23456          ! コメント文（コンパイラに無視される・7 カラム目が見やすいように入れた）
      program Q1                ! program 文（宣言文・省略可・ファイル名と同名がお勧め）
      real a, b, result         ! 型宣言文（暗黙の型宣言〔後述〕に従う場合省略可）

      a=2.0                     ! 変数 a に 2.0 を代入・代入文という・これ以降実行文
      b=3.0                     ! 変数 b に 3.0 を代入

      result=a+b            ! 変数 a と変数 b の値を足し合わせ，結果を変数 result に代入
      write(6,*) a, '+', b, '=', result        ! 変数 a, b, result の値を画面に出力

      write(6,*)'プログラム終了．エンターキーを押して下さい．'          ! プログラムが
      read(5,*)             ! すぐに終了してしまわないようにエンターキーの入力待ち．

      stop                  ! 実行を終了させる実行文（end の直前の stop 文は省略可）
      end                   ! プログラムの終端を宣言（実行文に分類されている）．
```

Q.2 一次関数

```
c23456          ! コメント文（コンパイラに無視される・7 カラム目が見やすいように入れた）
      program Q2            ! program 文（宣言文・省略可・ファイル名と同名がお勧め）
      real x, y             ! 型宣言文（暗黙の型宣言〔後述〕に従う場合省略可）

      write(6,*)'x は？'     ! x を入力してもらう
      read(5,*) x           ! x を読み込む
      y=3*x+4               ! 計算する

      write(6,*)  y         ! 計算結果を画面に出力

      end                   ! プログラムの終端（実行文に分類されている）．
```

Q.3 三角形の面積（その２）

```
c23456
      program menseki
      implicit none          ! 暗黙の型宣言の機能を使わない場合これを宣言
      real area, s, a, b, c
      real ax, ay, bx, by, cx, cy ! 3点の座標を記憶する変数を宣言

      ax= 0.0                      ! 3点の座標を与える
      ay= 0.0
      bx=-1.0
      by= 2.0
      cx=-3.0
      cy= 0.0

c      write(6,*)'三辺の長さを入力して下さい. ' ! 消さずにコメントアウトした
c      read(5,*)a, b, c
      a=SQRT((bx-cx)*(bx-cx)+(by-cy)*(by-cy))  ! 座標値から辺の長さを計算
      b=SQRT((cx-ax)*(cx-ax)+(cy-ay)*(cy-ay))
      c=SQRT((ax-bx)*(ax-bx)+(ay-by)*(ay-by))

      s=(a+b+c)/2.0
      area=SQRT(s*(s-a)*(s-b)*(s-c))
      write(6,*)'面積は', area, 'です. '

      stop
      end
```

Q.3 三角形の面積（その２／行列式による解法）

```
c23456
      program menseki_by_determinant
      implicit none
      real area, det
      real ax, ay, bx, by, cx, cy ! 3点の座標を記憶する変数を宣言

      ax=0.0                       ! 3点の座標を与える
      ay=0.0
      bx=-1.0
      by=2.0
      cx=-3.0
      cy=0.0
c ax=0, ay=0（原点）より
c     | bx cx | = ベクトル (bx, by), (cx, cy) の張る平行四辺形の面積
c     | by cy |
      det=bx*cy-by*cx
      area=ABS(det)/2.0 ! どちらを b にするかで符号が変わるので
                        ! 念のため絶対値 ABS() をとる
      write(6,*)'面積は', area, 'です. '
      end
```

2　変数の種類と変数の使い方

変数 (variable) には，次のような種類があります．それぞれ，記憶の形式（数値を二進数で表すときのやり方）や記憶に用いられるメモリの量が違います．太字はこの授業でも取り上げる基本的な型です．[詳細は教科書（CDROM¥text¥main.pdf）5 章 p.33–35]

1. **整数型**（integer）：2，-5 など整数値を記憶するためのもの．計算によって生じる誤差が無いが，扱える範囲は狭い（-2147483648$=-2^{31}$〜2147483647$=2^{31}-1$）．一つの数値に対して 4 バイト (32 ビット）用いる．

2. **単精度実数型**（real）：3.1416 や 6.02×10^{23} のように小数点以下も記憶する．有効数字は 7 桁程度であり，必ず誤差を伴うことに注意．範囲は $\pm1.0\times10^{38}$〜$\pm1.0\times10^{-37}$．一つの数値に対して 4 バイト (32 ビット，符号部 1，仮数部 23，指数部 8 ビット）．

3. 倍精度実数型（double precision，real*8）：単精度実数型と同様に実数値を記憶する．違いは精度が高く（有効数字 14 桁程度），かつ，記憶できる範囲が広いこと（$\pm1.0\times10^{308}$〜$\pm1.0\times10^{-307}$）．一つの数値に対して 8 バイト (64 ビット，符号部 1，仮数部 52，指数部 11 ビット）．

4. **文字型**（character*n）：'Hello, world!' のような文字（列）を記憶する．n は記憶する最大文字数．ただし，全角文字 1 文字は 2 文字と数える．

5. **論理型**（logical）真 (.TRUE.）と偽（.FALSE.）の二つの値をとる論理演算用変数．1 バイト

6. 単精度複素数型（complex）：$1.5+3.0i$　など実部と虚部の二つの実数値を組にして記憶．実部，虚部各 4 バイト，計 8 バイト．Fortran のプログラム中で複素型定数を用いる場合は下の例に示すように丸括弧で囲って実部と虚部を記す．

7. 倍精度複素数型（complex*16）：実部と虚部のそれぞれを倍精度実数型とした複素数．実部，虚部各 8 バイト，計 16 バイト．

型宣言の例

```
program sample
implicit none     ! 「暗黙の型宣言」を禁止する
integer icnt      ! icnt という整数型変数を宣言
real val, sum     ! val と sum という二つの単精度実数型変数を宣言
complex x         ! x という単精度複素型変数を宣言
character*6 a     ! a という半角文字 6 文字まで憶えられる文字型変数を宣言

icnt=0          ! 整数型変数 icnt に 0 を代入する
val=1.0         ! 単精度実数型変数 val に 1.0 を代入する
                !    （実数型変数には必ず小数点を付けた数値を与える）
x=(1.5,3.0)     ! 単精度複素数型変数 x に 1.5+3.0i を代入する
a='T66160'      ! 文字型変数 a に 「T66160」という文字を代入する
.....
end
```

3 暗黙の型宣言

型宣言をしない場合，変数名の一文字目が i〜n のときには整数型変数，それ以外の変数は単精度実数型と見なされます．この機能を「暗黙の型宣言 (default implied typing)」と呼びます．また，Fortran では型宣言をする場合でも変数名には上記の規則を適用するのが普通です．[教科書 p.38]

なお，暗黙の型宣言を用いると，変数名のミスタイプがコンパイル時に検出できなくなるため，致命的なバグの原因になります．例のように implicit none と宣言し「暗黙の型宣言」を禁止してから，プログラムで使う変数をすべて型宣言することを強く勧めます．implicit none を宣言した場合，型宣言していない変数を使用すると次のようなエラーメッセージを出力して，コンパイルを中止します．

implicit none 宣言によって変数名のミスタイプを検出した例

```
c23456
      program menseki
      implicit none                          ! 暗黙の型宣言を無効にする宣言（推奨）
      real area, s, a, b, c

      write(6,*)'三辺の長さを入力して下さい．'
      read(5,*)a, b, c

      s=(a+b+c)/2.0
      are=sqrt(s*(s-a)*(s-b)*(s-c))      ! area を are にミスタイプ
      write(6,*)'面積は', area, 'です．'

      write(6,*)'プログラムを終了します．エンターキーを押してください．'
      read(5,*)

      stop
      end
```

```
C:¥work¥T66160¥20160426>f77 menseki.f -o menseki
menseki.f: In program ‘menseki’:
menseki.f:10:                          ← 10 行目の are が問題
         are=sqrt(s*(s-a)*(s-b)*(s-c))
         ^                               ←（^で問題箇所を示している）
Invalid declaration of or reference to symbol ‘are’ at (^) [initially seen at (^)]
            理由は「シンボル are の無効な宣言 (declaration) あるいは参照 (reference)」
```

4 演算子，式

Fortran 言語では，変数や定数を算術演算子 (arithmetic operator) でつなげて一つの値を表す式を算術式 (arithmetic expression) と呼びます．例えば，$ax^2 + bx + c$ は a*x**2+b*x+c のような算術式で書くことができます．Fortran の算術演算子には次の 5 種類があります．[教科書 6.1 章，p.45–48]

算術式を用いるときには以下の点に注意してください．

1) 算術演算子を 2 つ以上続けることはできない．x=a*-3 (NG) → x=a*(-3)(OK)

2) 上表中の優先順位に従って計算される．例えば a+b*c**d は $a + \{b \times (c^d)\}$ となる．

表 1: Fortran の算術演算子

演算子	意味	例	優先順位
**	べき乗 (a^b)	x=a**b	1
*	乗算	x=a*b	2
/	除算	x=a/b	2
+	加算，正符号	x=a+b，x=+c	3
-	減算，負符号	x=a-b，x=-c	3

3) 同じ優先度であるときは左から計算される．a/b*c は $(a/b)c = ac/b$，a/b/c は $(a/b)/c = a/(bc)$ である．

4) 3) の例外としてべき乗は右から計算される．例えば a**b**c は $a^{(b^c)}$ であり，$(a^b)^c$ ではない．

5) それぞれの演算ごとに適当な型の一致が行われる．型の一致とは演算子の左右の変数や定数の型に従って演算結果の型を合わせることで，左右の型が同じならばその型，左右の型が違う場合には表現力の高い方へ合わせる．例えば，x=3/4*5.0 ならば $x = (3/4) \times 5.0 = 0 \times 5.0 = 0.0$，x=3*5.0/4 ならば $x = (3 \times 5.0)/4 = 15.0/4 = 3.75$ となる．

6) 演算順序に自信がないときには**丸括弧 () を使って演算順序を明示**しましょう（丸括弧のみを何重にも使用できます・{ }，[ ] は使えません）．

5 write 文と read 文

プログラムから計算結果やメッセージを出力するには write 文，プログラムにデータを入力するには read 文を用います．これらの文は**入出力文**と呼ばれます．[より詳しい説明は教科書 2 章，p.12–14 参照]

write(6,*)，read(5,*) の括弧の中は制御並びと呼ばれ，括弧内に　**入出力の装置番号 (unit number)**，**書式指定子 (format specification)** の順に書きます．装置番号は 5（キーボード・read 文専用），6（画面・write 文専用）があらかじめ定義されているので，特別の準備なしにそのまま使えます．Gnu f77 では他に装置番号 0 （画面・write 文専用・標準エラー出力）が割り当てられています．

書式指定子 (format specification) は，たとえば,「左から 5 カラムまでに右詰で数値を出力」のように入出力の形式（**書式**）を指定するもの（詳しい説明は後日）で，*は書式を特に指定せずに,「コンピュータにお任せ」するという意味です．

6 練習問題・宿題

1. ある地点において，半径 r=2.10 cm の球形の錘を質量を無視できる長さ ℓ=103.20 cm の針金の先に付けて吊し，振動させたところ，その周期 T は 2.0588 s であった．この地点の重力加速度 g を次式に従って算定せよ．

$$g = \frac{4\pi^2}{T^2}\left\{(\ell + r) + \frac{2}{5}\frac{r^2}{\ell + r}\right\}$$

ヒント：このような小数点以下も必要な計算では，実数型変数を使います．同様に，定数項も 2.0/5.0 のように書き，2/5 と書いてはいけません（2.0/5.0=2/5.0=2.0/5=0.4 ですが，2/5=0 となります）．

2. 三角形の二辺の長さと間の角を入力して面積を求めるプログラムを作成せよ．

ヒント：sin の計算は組込関数 sin を使いますが，与える角度がラジアン単位であることに注意して下さい．例えば，角度が変数 t に度単位で入力されているときには，sin(t*3.1415927/180.0) のようにラジアン単位に変換しながら組込関数を使ってください．(詳しい説明は CDROM の教科書（¥text¥main.pdf）の p.50 を参照．)

3. 西暦 y 年 m 月 d 日の曜日は

$$y + [y/4] - [y/100] + [y/400] + [(13 \times m + 8)/5] + d$$

を 7 で割った余りで与えられる（ツェラー (Zellar) の公式）．ここで，$[x]$ はガウスの記号であり，x を越えない最大の整数値（簡単に言えば実数値から小数点以下を取り除いた整数部分）を表す．また，1 月は前年の 13 月，2 月は前年の 14 月とし，余り 0 は日曜日，1 は月曜日,... を意味するものとする．

この公式に基づいて，次のような出力をする曜日計算プログラムを作成し，自分の誕生日の曜日を調べよ．

曜日計算プログラムの実行例

```
C:¥work¥T66160¥20170424>f77 Zellar.f -o Zellar

C:¥work¥T66160¥20170424>Zellar
 年，月，日を入力してください．                ←入力を要求するメッセージを出力
 ただし，1 月は前年の 13 月，2 月は 14 月とします ←長いので write 文を二度使った
2017, 4,24   ←年月日を入力した（チェックに自明な値を使う）
 曜日は  1 です．(0 は日曜日，1 は月曜日 ... です．) ←計算結果が出力された
```

ヒント：ガウスの記号の部分は，Fortran では二つの整数型のわり算の結果が小数点以下を切り捨てた整数値になることを利用すれば簡単に計算できます。すなわち，整数型同士の割り算を行えば自動的にガウスの記号の計算になります。(下記プログラムの isho=の部分を参照）余りの計算も，このことを利用して二つの整数型のわり算を使って行うことができます（下記プログラムの iamari=の部分を参照）．余りの計算には組込関数の mod() 関数を使うこともできます．

整数型変数・整数型定数を用いた商，余りの計算

```
c23456
      integer iy, ishou, iamari
      iy=2014
      ishou=iy/4              ! 2014/4=503.5 でなく 503 になる
      iamari=iy-iy/4*4        ! 2014/4 の余りの計算
                              ! ∵ 2014-2014/4*4=2014-503*4=2014-2012=2
      write(6,*)ishou, iamari
      end
```

質問があれば配布した紙の下の方に書いておいて下さい．