JAPLA 研究会資料 2014/6/14

ＪのGerund(動詞列)を使う構文
－Keplerの第３法則の計算を例として－

西川　利男

　Ｊには、他の言語ならば、プログラムを組まなくてはならない処理を数行のコーディングで行えるGerund(動詞列)という構文がある。ちょうど最近、読んでいた本にKeplerの第３法則の計算があった。これを例として、Ｊのgerund構文の使い方について、まとめてみた。

１．Keplerの第３法則

　これは、次のようなものである。

表2-1　惑星の公転周期と太陽からの平均距離
（有効数字4桁）

惑　星	公転周期（地球の一年を単位とする）	太陽からの平均距離（太陽から地球までの平均距離を単位とする）
水　星	0.2408	0.3871
金　星	0.6152	0.7234
地　球	1	1
火　星	1.881	1.524
木　星	11.86	5.201
土　星	29.50	9.545
天王星	84.02	19.55
海王星	164.8	30.11
冥王星	248.5	39.54

を単位にして記されています。太陽系の一番内側にある水星は、地球の時間でいうと八八日（約〇・二四年）で太陽の周りを一周しますが、一番外側の冥王星は約二四八年もかかります。

　さらに距離については次のように考えます。コペルニクスの宇宙像のように惑星の軌道が円軌道なら、太陽から惑星までの距離は常に一定ですが、ケプラーの楕円軌道では、太陽までの距離は軌道上の位置によって異なります。そこで、「一番長い半径」と「一番短い半径」の平均をとって、太陽-惑星間の平均距離とします。各惑星についてのこの平均距離が、表2-1の右の欄に示されています。値は、太陽から地球までの平均距離（約一億五〇〇〇万km）を単位にして記述されています。太陽からの平均距離は、一番内側の水星で地球の約三九％ですが、一番外側の冥王星では約四〇倍にもなります。

　各惑星について、「左の欄の値の二乗」を「右の欄の値の三乗」で割ったとき、どのくらい一に近くなるかを、チェックしてみてください。私自身はこれを最初にチェックしたとき、ケプラーの努力に思いを馳せて、感動を覚えました。ケプラーは、土星までの六つの惑星に関する多数の数値群を丹念に調べあげ、一〇年を費やして、公転周期と平均距離の関係を発見したわけです。「干草の山のなかから針一本を探し出す」という英語の表現を思い出しました。ケプラー自身も、「天文学を志した当初からの年来の望みが達せられた」と述懐しています。

　の欄に、各惑星の公転周期が地球の一年（地球の[illegible]

米沢富美子「人物で語る物理入門」（上）p. 35-37, 岩波新書(2005)

２．ＪのGerund（＝動詞列）を使う構文－１

・ＪのGerundとは、いくつかの動詞をtie(`)でつないで作る動詞列のデータである。
　このままでは、実行できない。
　（決して、英文法でいう動名詞と同じであるなどとして理解しないほうがよい。）
・これを実行するには、機能に応じた、Ｊのプリミティブの組み合わせ(`:nや /.や/.や@.)をGerundの後ろに付けた次のような構文で行う。

　具体的な例で示す。
(1) 動詞列と値の列との演算を行って、値を配列として返す。

```
     +: ` *: ` %: (`:0) 1 2 3 4 5
2        4        6  8       10
1        4        9 16       25
1 1.41421 1.73205  2 2.23607
```

(2)　動詞列と値の列との演算を行った結果から、斜め(/. Oblique)の値を取り出す。

```
   +: ` *: ` %: /. 1 2 3 4 5
      2
      4
1.73205
      8
     25
```

(3) 配列でなく値の並び（リスト）として取り出すには、つぎのようにすればよい。

```
   ,  @(+: ` *: ` %: /.) 1 2 3 4 5
2 4 1.73205 8 25
```

　まだ、他にGerund（＝動詞列）を使ったいろいろの構文があるが、
後の節４、５に示す。

３．Kepler の第３法則の計算

Gerund の構文を使って、Kepler の計算を行ってみよう。

プログラムはつぎのようになる。

```
NB. Kepler Calc.
Planet =: ". ] ;._2 (0 : 0)
 0.2408 0.3871
 0.6152 0.7234
 1      1
 1.881  1.524
11.86   5.201
29.50   9.545
84.02  19.55
164.8  30.11
248.5  39.54
)
square =: *:
cube =: 3 : 'y. ^ 3'
kepler =: , @(square ` cube /.)
```

実行したようすはつぎのようになる。

```
   Planet
0.2408 0.3871
0.6152 0.7234
     1      1
 1.881  1.524
 11.86  5.201
  29.5  9.545
 84.02  19.55
 164.8  30.11
 248.5  39.54

   kepler "(1) Planet
0.0579846 0.0580055
 0.378471  0.378561
        1         1
  3.53816   3.53961
   140.66   140.689
   870.25   869.617
  7059.36   7472.06
    27159   27298.1
  61752.3   61817.3

   %/"(1) kepler "(1) Planet
0.99964 0.999763 1 0.999592 0.99979 1.00073 0.944768 0.994906 0.998948
```

４．Ｊの Gerund（＝動詞列）を使う構文－２

まだ、他にもいろいろな Gerund を使う構文がある。

(4) 動詞を次々と挿入する。

```
   + ` * (`:3) 1 2 3 4 5
47
   + ` * / 1 2 3 4 5
47
   1 + 2 * 3 + 4 * 5
47
```

このとき、Ｊでは右から演算されることに注意！
以下のように演算されて、値が得られる。

```
   1 +  2 *  3 +  4 * 5
            3 + 20
        2 * 23
   1 + 46
  47
```

(5) 動詞列を fork や hook として実行する。

```
   + ` * ` - (`:6) 1 2 3 4 5
_1 _4 _9 _16 _25

   (+ * -) 1 = (+1) * (-1)
_1
   (+ * -) 2 = (+2) * (-2)
_4
   (+ * -) 3 = (+3) * (-3)
_9
   (+ * -) 4 = (+4) * (-4)
_16
   (+ * -) 5 = (+5) * (-5)
_25
```

５．Ｊの Gerund（＝動詞列）を使う構文－３

(6) 次は、@.以下の条件によって、動詞列から１つの動詞を選択して実行する。

```
   2 (+ ` ^)@.(<) 3  NB. 2<3 => true(1) => 2 ^ 3
8
   3 (+ ` ^)@.(<) 2  NB. 3>2 => false(0) => 2 + 3
5
```

(7) 次のような再帰的定義、fact 階乗も Gerund 構文で可能である。

```
   fact =: 1:`[] * $:@<:)@.*
   fact 5
120
```

(補注) 次でもよい

```
square=: *:
cube=: ^&3
kepler=: ,@(square `  (cube) /.)
```