JAPLA研究会資料　2022/8/1

Ｊで数式を美しく表わしてみよう- 改良版
自然数の和

西川　利男

コロナ禍のため、毎日家にいることをよぎなくされている。ごく最近つぎのような数学の本を入手した。[1]

[1]　横山明日希「数式図鑑－美しく、役に立つ科学の宝石箱」
　　ブルーバックス　B-2178 講談社(2022)

Ｊで難しい計算を行うのはもちろんだが、計算途中や結果をきれいに表示し、印刷することにもＪは極めて有効である。

数式図鑑の第１ページに下のようなきれいな計算が載っている。つれづれなるままに、これをＪでプログラムをやってみた。こんなもの、たいしたことないとたかをくくっていたが、なかなかてこずった。[2]

[2] 西川利男、「Ｊで数式を美しく表わしてみよう－自然数の和」
　　JAPLA研究会資料　2022/3/21 柏パレット　ルームＧ

その後、プログラムを見直しが、あまりにお粗末なコーディングでありったので
もっとすっきりしたものに改良した。

文献[1]には、この数式のなぜかの説明が次のページのようにある。

これにしたがって、コーディングをし直した。

和が美しく図形のように連なる感動

自然数の和

$$1+2=3$$
$$4+5+6=7+8$$
$$9+10+11+12=13+14+15$$

自然数を並べていくだけで、美しい式が作れることをご存じでしょうか？　２つの連続する自然数を足すと次の自然数に、３つの連続する自然数を足すと、次に連続する２つの自然数の和に……というように、左辺の項数と右辺の項数の差が１になった自然数の式が連続して現れます。どうしてこのような式が成立するのかを考えてみましょう。

よく見ると、いちばん左の項は１，４，９と、平方数（整数の２乗となっている数）になっていることがわかります。

２行目と３行目の式を例に考えると、$4=2+2$,
$9=3+3+3$となるので、

$$\underline{2+2}+5+6=7+8$$
$$\underline{3+3+3}+10+11+12=13+14+15$$

と書き換えることができます。ここで上の式では２つの2を５，６にそれぞれ足すと７，８になり、下の式では3つの３を10，11，12にそれぞれ足すと13，14，15になります。つまり、右辺と同じ式になることがわかりますね。

では、もう少し一般化して考えてみましょう。
冒頭の式は、いちばん左の項を$a\times a$の平方数として、

$$(a\times a)+(a\times a+1)+(a\times a+2)+\cdots+(a\times a+a)$$
$$=(a\times a+a+1)+(a\times a+a+2)+\cdots+(a\times a+a+a)$$

と表すことができます。左辺の項数は全部で$(a+1)$個、右辺の項数は全部でa個です。ここで、いちばん左の項を$(a\times a)=(a+a+a+\cdots+a)$と変形して、各項にそれぞれ$a$を足していくと、左辺は、

$$(a\times a+a+1)+(a\times a+a+2)+\cdots+(a\times a+a+a)$$

となります。この式は、右辺とまったく同じ式になりますね。このような式変形をしてみることで、隠された美しい法則性が見えてくることが数式にはよくあります。

１．Ｊのプログラム化への考え方

最初に強調したいこの課題の基本の考え方のポイントは

「これは数値の計算ではなく、数字の文字列の結合などの処理である。」

まずは、具体的に上の説明通りにやってみよう。

まず、先頭の値４の平方根をとる。
４　＝　２＾２
つまり２＋２
１から２までの自然数を生成する。
１、２
それを、先頭の値に加える
５、６
これの前に先頭の値４を追加する
４、５、６
これが左辺になる。
右辺は、左辺の最後の数の続けて、２項つづける。
７、８
これから
４　＋　５　＋　６　＝　７　＋　８
のように数式が完成する。
　また、このように左辺は３つの項、右辺は２つの項
つまり、右辺の項の数は左辺の項の数より１少ない、ことにも注意。

同じように、次の式をやってみる。
先頭の値９の平方根をとる。
　９　＝　３＾２
つまり
　９　＝　３　＋　３　＋　３
１から３までの自然数を生成する。
１、２、３
それを、先頭の値に加える
１０、１１、１２
これの前に先頭の値９を追加する
９、１０、１１、１２
これが左辺になる。
右辺は、左辺の最後の数の続けて、３項つづける。
１３、１４、１５
これから
９＋１０＋１１＋１２　＝　１３＋１４＋１５

のよう数式が完成する。

２　Ｊのプログラム
２.１　Ｊプログラムを１ステップずつ作る
　上の考え方を一般的な関数として、次のように定義する。
左辺を作る関数を func0 、右辺を作る関数を func1 とした。数式の最初の値を引数とする。

```
func0 =: 3 : 0
J =. y.
(*: J) + (i. >: J)
)
func1 =: 3 : 0
J =. y.
(>: J) + (*: J) + (i. J)
)

   func0 2
4 5 6
   func1 2
7 8

   func0 3
9 10 11 12
   func1 3
13 14 15
```

のように実行される。

　上の定義 func0, func1 を使って
これをもとに、式の形で表示するようにする。
　ここで、値をまとめて扱うＪのボックス化という機能が大切である。
　例えば、11 番目の式を作ってみよう。
定義 func0, func1 で得られた数値を<″(0)により、それぞれボックス化する。

```
P =: <″(0) func0 11
Q =: <″(0) func1 11
```

　これらのボックスの中の数値を″: L:0 により文字化する。
そして、ボックス内で数値の文字列の間に文字 ‘+’ を挿入する。
次に>によりボックスをはずしてから、, により一行の文字列とする。
最後に最終の文字を落とす。

　このような操作を一行で行い、それぞれ PP, QQ とする。

```
PP =: }: , > (,&'+') L:0 ″: L:0 P

QQ =: }: , > (,&'+') L:0 ″: L:0 Q

NB. ================================
P11 =: <″(0) func0 11
Q11 =: <″(0) func1 11

PP11 =: }: , > (,&'+') L:0 ″: L:0 P11
QQ11 =: }: , > (,&'+') L:0 ″: L:0 Q11
```

２.２　Ｊプログラム funct として定義する

```
funct =: 3 : 0
PN =. <″(0) func0 y.
QN =. <″(0) func1 y.
PPN =. }: , > (,&'+') L:0 ″: L:0 PN
QQN =. }: , > (,&'+') L:0 ″: L:0 QN
(PPN ,'=', QQN) -. ' ' NB. revised for funct 3
)
```

３．１　プログラムを１つずつ実行

```
   funct 3
9+10+11+12=13+14+15
   funct 4
16+17+18+19+20=21+22+23+24
   funct 5
```

```
25+26+27+28+29+30=31+32+33+34+35
   funct 6
36+37+38+39+40+41+42=43+44+45+46+47+48
   funct 7
49+50+51+52+53+54+55+56=57+58+59+60+61+62+63
   funct 8
64+65+66+67+68+69+70+71+72=73+74+75+76+77+78+79+80
   funct 9
81+82+83+84+85+86+87+88+89+90=91+92+93+94+95+96+97+98+99
   funct 10
100+101+102+103+104+105+106+107+108+109+110=111+112+113+114+115+116+117+118+119+120
```

３．２　最終プログラム

最終のプログラムは、文字イコール‘=’を中心として、山形の一覧式をつくることにある。文字イコール‘=’の位置をきめて、数式の前にスペースを置いて、中央になるようにする。

```
NB. == newly revised 2022/4/1 ================================================
run =: 3 : 0
60 run y.
:
center =. x.
N =. y.
i =. 1
while. i <: N
   do.
      res =. funct i
      keta =. res i. '='
      wr ((center-keta)#' '), res
      i =. i + 1
   end.
'*** end ***'
```

４　プログラムの実行

左引数にスペースの位置、右引数にどこまで表示するかを指定して実行する。

```
   30 run 9
                                  1+2=3
                               4+5+6=7+8
                          9+10+11+12=13+14+15
                     16+17+18+19+20=21+22+23+24
                25+26+27+28+29+30=31+32+33+34+35
           36+37+38+39+40+41+42=43+44+45+46+47+48
      49+50+51+52+53+54+55+56=57+58+59+60+61+62+63
   64+65+66+67+68+69+70+71+72=73+74+75+76+77+78+79+80
81+82+83+84+85+86+87+88+89+90=91+92+93+94+95+96+97+98+99
*** end ***
```