

## 小数点付きの 基数変換 #dot (第2報)

SHARP.dot: N-base Transformation with Decimal Point (Seq. 2)

中野嘉弘 (札幌市南区・85才)

NAKANO Yoshihiro (Sapporo, Japan)

FAX 専 011-588-3354      yoshihiro@river.ocn.ne.jp

小数点付きの N進法基数変換 の第2報である。  
頭の体操から出発して、J言語の実際面で役に立つ何かへ発展させたい！

### 0. は し が き

先月の JASPLA 例会の同名の報告 (文献1) の続報である。

J言語の組み込み関数に、基数変換 (10進数  $\leftrightarrow$  N進数) の為のもの #.(Base) と #:(Antibase) が用意されている。  
ただし、対象は正の整数だけらしい。と云うのは、実数 (小数点付き) の計算例は、少なくとも、私は拝見したことは無い。

前報について、西川会長と志村幹事長?から、それぞれ、早速のコメントが届いた。

#### 1) 西川コメント FAX (文献2) :

「2進数の 1.0011011 の 17倍は 10100.1001011 で正解でしょうか? の解ならば 16倍 (4桁右シフト) と 1倍 の足し算で直ぐ見付かるよ。」 御名答ですね。  
ただし、知恵袋の回答者の二人 (内、一人は自称・小学生) の回答は、いずれも、西川コメントと同じ方法であった。 どうも、これが定番なのだ。

ここまでは出発点! 果たして、その後の西川FAX (文献2-a) では  
「2進法の小数点の問題、いまいちポイントがはっきりしません。 頭の体操なのか? コンピュータ内部の問題なのか?」 が到来した。 こりゃ、何とかせにゃならん。

(中野) なるほど! 例えば、掛け算では無くして割り算ならどうする?  
さらに、自分としては、一般に N進数の話題のつもりだったが?

#### 2) 志村メール (文献3) :

「中野関数 antidb を用いて 2 antidb 1.001001 をトライした。

しかし、右引数のデータ (小数点付き2進数) を 10進数 に直せなかった。」

■ 中野の返信 (文献4) : J言語の prime関数 だけを用いても

0.5 #:|:1.001001 から、直ぐに

10進数で 1.14063 と返答されますので、今回の小生の関数 antidb では

整数部の処理は省いてあります。 今後、暇を見付けたら拡張します。  
(とりあえず、多謝！)

とにかく、こんな話題があったので、続報を書く事にしました。  
前報の中野関数 `decibase` や `antidb` 等々は、話題を引き出す為の前座であり  
ます、Hi！ 拡張しても、それほど意味があるとは思えないが？

J言語の組み込み関数 `#.` や `#:` の解説 (HELP などを含め) には、小数点付き  
の話題を見た経験が無いので、とにかく、会友の皆様の興味を引き出したいのが、目的  
です。 「コロンブスの卵」になれるか、どうかな？

志村さん等の「J 便覧 Quick Reference」など、Tutorial の例題にでも、追加されれば  
嬉しい。

以下、多少の「おさらい」を兼ねて、補足説明します。

## 1. 自作のプログラムでは

形だけ J 言語による愚直なプログラムを 2 つ作った。 Script は末尾に。

1) 10 進数から N 進数へ `decibase`

2) N 進数から 10 進数へ `antidb`

例題：

● 与数 `0.0011011` 等 (`bin 2 進数`) を 10 進数に戻してみます。

`2 antidb 0.001001 -> 0.140625` (`dec 10 進数`) です。

◎ これを 2 進数に直します。

`(2,6) decibase 0.140625 -> 0.001001`

## 2. J 言語の組み込み関数 `#.` `#:` でも

出来るか否か？トライして見たら、出来ちゃった！ 元来、これを云いたかったのだ。

● 2 進数の小数部を 10 進数 に直す。 2 の 逆数は 0.5 です。

`0.5 #. 001 -> 1`

`0.5 #. 0010 -> 0.5`

`0.5 #. 00100 -> 0.25`

`0.5 #. 001000. -> 0.125`

`0.5 #. |. 0.00100 -> 0.125`

`0.5 #. |. 0.00100 -> 0.25`

`0.5 #. |. 0.001001 -> 0.140625` (`dec 10 進数`)

`0.5 #. |. 1.001001 -> 1.140625` (`dec 10 進数`)

【要注意】 小数点付近を、離さずに `0.0` の如く、密着すれば

`0.5 #. |. 0.001001 -> 0.28125` (`dec 10 進数`)

結果は、その上の例の 2 倍 が返される。

`0.5 #. |. 1.001001 -> 1.28125` (`dec 10 進数`)

データ入力の書き方によっては、2 倍も間違った結果になる。

● 整数部 と 小数部 が混在の場合の例： 与数 `20.5859`

整数部は `(5 # 2) #: 20 -> 10100`

小数部は `<. (9 # 2) #: 0.5859 * 2^9 -> 100101011`

両者を加えて、 10100.100101011 (答: bin 2進数)  
出来ました。

2進法を一般の N進法に換えるのも簡単。前報(文献1)から再録して置く。

### 3. N進法へ拡張

8進 OCTAL と 16進 HEX で例示しよう。

1) 8進法 与数は 実数 20.5859  
(2#8)#: 20 → 24  
<.(7#8)#: 0.5859 \* 8^7 → 4537661  
答は 2 4 . 4 5 3 7 6 6 1

検算(整数部は省く)  
+/(4537661) \* (8^(\_1)) \* (1+i.7) → 0.5859  
小数点以下 6桁 では → 0.585899

2) 16進法 与数は 実数 20.5859

文字列 ABCD =: '0123456789ABCDEF.' を用意して置く。  
整数 19, 29, 129, 229, 329 での例:

(4#16)#: 19 → 0 0 1 3  
((4#16)#: 29) { ABCD → 001D  
((4#16)#: 129) { ABCD → 0081  
((4#16)#: 229) { ABCD → 00E5  
((4#16)#: 329) { ABCD → 0149

● 整数部 (2#16)#: 20 → 1 4  
1 4 { ABCD → 14 (hex)

小数部 (<.(4#16)#: (0.5859)\*16^4) → 9 5 15 13  
9 5 15 13 { ABCD → 95FD (hex)

まとめて 20.5859 (dec) → 14.95FD (hex)

検算 '14.95FD' i. ~ ABCD → 14 16 9 5 15 13  
14 16 9 5 15 13 { ABCD → 14.95FD

16#.1 4 (hex) → 20 (dec)  
+/(9 5 15 13) \* 16^(\_1) \* (1+i.4) → 0.585892 (dec)

前の 2. 節の 2進法の冒頭の時とは多少、変わった解法であるが、色々なやり方があるのは当たり前で、それらを狙ったのが、この小稿の目的である。

### 4. 小数付け基数変換の意味

小数付け基数変換と云うても、要は(例えば2進法の)表示のシフトが主要な事で、それに若干の加減算をすれば済む問題と考えられるかもしれぬ。  
与数の17倍如き、掛け算ならば、簡単かもしれぬが、割り算では、そうは行かぬ。

例えば、10分の1にする時、例えて2進表示ならば、どうでしょうか？

$$1/10 = 1/16 + 1/32 + 1/64 - 1/128 - 1/256 + 1/512 = 0.099609375....$$

$$= 0.1 - 0.000390625.....$$

$$= \text{誤差 } 0.0004 \text{ 即ち } 0.4\% \text{ であるから}$$

2進表示の幾桁づつかの左右のシフトを6回以上は反復せにやらぬ。

これは2進表示の為であるから、先ず10進表示でやれば、1回ですむ。その後に、例えば2進表示に変換すれば良しとなろう。と考えられよう。

実は、その事を実践して見せたのが、この小稿なのである。

当初は、その為の(中野)関数如きを作成してみたが、実はJ言語の prime である #. や #: を利用しても、簡単に同じ事が出来る事に気付いたのであった。

志村メールに対する中野の返信メールがそれである。

答が簡単過ぎて、「問題のポイントは何か？」と西川FAX(文献2-a)の如き心配がもたらされたので、この第4節で意味を強調したのだ。

## 5. 他の数学ソフトで可能例があるか？

少数点付きの基数変換は、J言語で出来る事を縷縷しめしたが、同じ事が、他の数学ソフト、Mathematica や Maple V などではどうなっているか？ 何度か調べたが、関係演算例が見つからぬ(御調査を請う)。

ところが、国産の数学ソフト「カルキング Calkinng」では、ちゃんと出来るように設計されているのだ(文献5)。例示しよう。無指定は10進数(dec)である。また bin oct hex 等は、実際は ( ) の外の 小添字(2, 8, 16 等)である。

$$10.2929 = (1010.01001011000) \text{ bin}$$

$$(1010.0100101011) \text{ bin} = 10.2919921875$$

$$(1.0011011) \text{ bin} = 1.2109375$$

$$(1.0011011) \text{ bin} * 17 = 20.5859375$$

$$20.5859375 = (10100.1001011) \text{ bin} \quad \text{第0節 はしがき の 西川解と同じ。}$$

「カルキング Calkinng」では、基数として2進、8進、10進、16進 の他 30 までの任意の正整数が可能である。ただし、与数は正の整数と正の小数とする。この数学ソフト「カルキング」の第1版は1994年であるが、設計が米国産のに似ているので、調べたところ、上記の点では、日本製の方が勝っているようだ。

(小生の Mathcad では不可能！)

兎に角、国産ソフトで、「小数点付き基数変換」可能のものがある事だけは、主張して置きたい。

## 6. J 言語の 基数変換 の優越性

これは、J の tutorials の講師の方に、力説をお忘れ無く、お願いしたい。前節の「カルキング」でも基数変換の対象の与数は「正の数」との条件が付いた。然らば、J言語ではどうか？ 「負の数」でも可能である。但し、返されるのは「補数」である。以下、それらを例示しよう。

$$1) \quad \text{整数 正} \quad (9\#2)\#:3 \rightarrow 000000011$$

$$\text{負} \quad (9\#2)\#:_3 \rightarrow 111111101$$

$$\text{少数部 正} \quad <.(9\#2)\#:0.3*2^9 \rightarrow 010011001$$

負 <. (9#2) #:  $_{0.3} * 2^9$  -> 1 0 1 1 0 0 1 1 0

2) 与数から 整数部 と 少数部 の分離法

0 1 #: 1.23 -> 1 0.23

0 1 #: 145.23 12.34 1.234 -> 145 0.23

12 0.34

1 0.234

1 | 12.34 -> 0.34

1 | \_1234 -> 0..66 互いに補数 (加えて 1)

3) 連続与数の分離法の例

0 1 10 100 1000 #: 123456789 -> 1234 0 4 56 789

## 7. む す び

こんな話は、余り見かけなかいと思うので、取りあえず御紹介します。

他の数学ソフト、Mathematica や Maple V などではどうなっているか？

何度か調べたが、関係演算例が見つからぬ。御支援を請う。

むしろ、姉妹言語 APL の方が、参考になった。

なんでも、矮小視せずに、トライして見るものですね。

## 文 献

1) 中野嘉弘「小数点付きの基数変換 # d o t」 JAPLA 2008/Nov/22, pp.4

2) 西川 F A X 「1 7 倍とは 1 6 倍 + 1 倍 なので・・・」 '08.11.19

- a) 西川 F A X 「2 進法の小数点の問題のポイントは？」 '08.11.30

3) 志村メール「JAPLA NOV 2008、中野先生 2b1.001001 も試してみて・・・」

<JAPLA@aplsoft.co.jp> Nov. 25, 2008 4:00 PM

4) 中野返信メール「J の prime だけ用いて可能ですので・・・」

<yoshihiro@river.ocn.ne.jp> 2008.11.29. 10:47

5) 「カルキング 8 ユーザーズガイド」 (株) シンプレクス 2007.8.6

サポートセンター TEL 042-783-0457、 FAX 042-783-0456

<http://www.simplex-soft.com>