２００２年1月２６日

Jのフレーズについて
＿＿＿＿ 第１回 ＿＿＿＿

慶応義塾大学理工学部
竹内寿一郎

第１章　はじめに

A 節．　用法

ここで使用するフレーズの先頭の 1 文字は品詞や機能によって決められる。
その 1 文字は次の表による。

記号	機能	記号	機能
a	副詞	m	単項動詞
c	接続詞	n	名詞
d	2 項動詞	v	曖昧項動詞

文字のうしろに付けられた数字は、節毎に与えられる番号なので名前としては重複する。よく使用されるフレーズの場合には特に名を付けられ、m2=:mean=:＋/%#のように 2 重に定義される。

定義	簡単な説明
n0=:i.6	ゼロから 6 個の非負の整数からなるリスト
m1=:^&3	3 乗
m2=:mean=:＋/%#	算術平均
d3=:$,:	y を形状 x でコピー。結果の形状は (x,$y) となる
m4=:<.@(0.5&＋)	まるめ。負の数と虚数に注意
m5=:=＋	実数テスト
m6=:(<0 _1)&C.	先頭と最後尾のアイテムの入れ替え
m7=:＋/\-:＋/\.&.|.	prefix 和は逆順 under の下での suffix 和に等しい

初めて J に触れる人は引数として、アトム、リスト、テーブル、より高階のアレーなどで試みたり、整数、負の数、実数、虚数などで試してみるとよい。

ここで、実行にあたって注意すべき点をいくつかあげておく。

●数字が続くとき、リストの切れ目に注意しなければならない。

```
   ^&3 y=:0 1 2 3 4
0 1 8 27 64
   ^&3 0 1 2 3 4
+-------------+
|^|&|3 0 1 2 3|
+-------------+
   ^&3(0 1 2 3 4)
0 1 8 27 64
```

リストを作用させるとき、区切りを付けねばならないこともある。

●切り上げ、切り捨て、四捨五入では、負の数や虚数の場合に注意！

```
   m4 1 2 1.4 1.5 _1.4 _1.5 1.5j1.5 _1.5j_1.5
1 2 1 2 _1 _1 2j1 _1j_2
```

とくに、虚数の場合のまるめは一体どうなっているのでしょう。考えてください。

●先頭と最後尾の交換

```
   m6 'ABCDabcd'
dBCDabcA

   m6"2 i.3 4 5
15 16 17 18 19
 5  6  7  8  9
10 11 12 13 14
 0  1  2  3  4

35 36 37 38 39
25 26 27 28 29
30 31 32 33 34
20 21 22 23 24

55 56 57 58 59
45 46 47 48 49
50 51 52 53 54
40 41 42 43 44
```

ランクを指定すると交換するアイテムが異なってくることに注意。

●アレーの平均をとる

```
   report=:?2 4 3$10
   report
1 7 4
5 2 0
6 6 9
3 5 8

0 0 5
6 0 3
0 4 6
5 9 8

   (mean;mean"2;mean"1)report
+-----------+--------------+------------------------------+
|0.5 3.5 4.5|3.75    5 5.25|       4 2.33333        7 5.33333|
|5.5   1 1.5|2.75 3.25  5.5|1.66667        3 3.33333 7.33333|
```

```
|  3   5 7.5|              |                                |
|  4   7   8|              |                                |
+-----------+--------------+--------------------------------+
```
ランクを指定すると平均をとる軸（？）が異なる。

●コピーの 2 項動詞 d3 は単項としても働く
```
   2 2 d3 1 2 3
1 2 3
1 2 3

1 2 3
1 2 3

   d3 1 2 3
1 2 3
1 2 3
1 2 3

1 2 3
1 2 3
1 2 3

  $2 3 d3 i.2 3
2 3 2 3

  $d3 i.2 3
2 3 4 5 2 3
```
単項のとき右引数がアレーのときはとてつもなく大きなアレーとなる。

●曖昧項動詞を避ける
```
   f=:[: :d3
   f i.2 3
|valence error: f
|       f i.2 3
   2 2 f i.2 3
0 1 2
3 4 5

0 1 2
3 4 5



0 1 2
3 4 5
```
注意：f=:[::d3 ではエラー、[:と:の間にスペースが必要。

```
0 1 2
3 4 5
```
d3 は曖昧項動詞であるが、f のように定義しておくと 2 項でしか使えなくなる。

●動詞、副詞、接続詞、名詞などの表現

```
   m7
+---------+--+------------------+
|+-----+-+|-:|+----------+--+--+|
||+-+-+|\||  ||+-----+--+|&.||.||
|||+|/|| ||  |||+-+-+|\.||  |  ||
||+-+-+| ||  ||||+|/||  ||  |  ||
|+-----+-+|  |||+-+-+|  ||  |  ||
|         |  ||+-----+--+|  |  ||
|         |  |+----------+--+--+|
+---------+--+------------------+
   9!:3(5)
   m7
+/\ -: +/\.&.|.
```
通常ボックスで表示されるが 9!:3(5) でリニアな表現に変更できる。

```
   9!:3(2 4 5)
   m7
+---------+--+------------------+
|+-----+-+|-:|+----------+--+--+|
||+-+-+|\||  ||+-----+--+|&.||.||
|||+|/|| ||  |||+-+-+|\.||  |  ||
||+-+-+| ||  ||||+|/||  ||  |  ||
|+-----+-+|  |||+-+-+|  ||  |  ||
|         |  ||+-----+--+|  |  ||
|         |  |+----------+--+--+|
+---------+--+------------------+
  +- \ ---- / ---- +
  +- -:
--+      +- \. --- / --- +
  +- &. -+- |.
+/\ -: +/\.&.|.
```
9!:3(2 4 5) とすると、ボックス、樹状、リニアの 3 つの表現を得る。

●動詞などの一部分を取り出して実行して確かめることができる

```
   x=:1 2 3 4 5 6
   +/\ x
1 3 6 10 15 21
   +/\. x
21 20 18 15 11 6
```

+/\を prefix 和、+/\. を suffix 和という。
これをボックスで確認する。

```
   <\ x
+-+---+-----+-------+---------+-----------+
|1|1 2|1 2 3|1 2 3 4|1 2 3 4 5|1 2 3 4 5 6|
+-+---+-----+-------+---------+-----------+
   <\. x
+-----------+---------+-------+-----+---+-+
|1 2 3 4 5 6|2 3 4 5 6|3 4 5 6|4 5 6|5 6|6|
+-----------+---------+-------+-----+---+-+
```

<\を prefix スキャンボックス、<\. を suffix スキャンボックスという。

●-:はマッチといい、全体が完全に等しいかどうか調べる

```
   m7 1 2 3 4 5 6
1
```

m7 は prefix 和が、逆順にしてから suffix 和をとりその結果を再び逆順にしたもの、に等しいことを示す。

●不慣れな関数は簡単な例で使ってみて、あとから複雑な例に適用してみるとよい

```
   s=:sort=:/:~
   numb=:3 1 4 1[char=:'cage'
   poem=:'i sing','of olaf',:'glad and big'
   (,.s numb);(,.s char);poem;(s poem);s"1 poem
+-+-+------------+------------+------------+
|1|a|i sing      |glad and big|       giins|
|1|c|of olaf     |i sing      |      affloo|
|3|e|glad and big|of olaf     |  aabddggiln|
|4|g|            |            |            |
+-+-+------------+------------+------------+
```

ランク 1 を指定すると、そのアイテムすなわちアトムに関してソートを行う。

●ユーティリティ

J でよく使われるいくつかの動詞や副詞など、J の実行コマンドに profile.ijs をつけておくと、J の開始時に自動的に読み込まれる。そうでなければ 0!:0<'profile.ijs' を実行すればよい。profile.ijs が読み込まれたかどうかは

```
   names ''
```

を実行してみるとよい。

●定義を固定する

定義	簡単な説明
n16=:m=:3 1 4,2 0 5,:1 4 1	3×3 の行列
d17=:ip=:+/ .*	内積、または行列積
m18=:L=:m&ip	線形変換関数

線形変換関数 L を定義して、それを f. で固定し、h とする。
その後 ip の定義を変更すると、L は変わるが、h は変わらないことに注意。

```
   L
+-----+-+--+
|3 1 4|&|ip|
|2 0 5| |  |
|1 4 1| |  |
+-----+-+--+
   L x=:1 2 5
25 27 14
   h=:'L'f.
   h x
25 27 14
   ip=:-/ .*
   L x
21 27 _2
   h x
25 27 14
```

なお、動詞の場合　h=:'L'f.　は' を省略して　h=:L f.　としてもよい。

B 節.　ロケールとスクリプトファイルの読み込み

各章各節のフレーズは次のようにJのシステムから読み込むことができる。

```
   load 'system\examples\phrases\phrd1.ijs'
```

ここで、ファイル名は phr 節章.ijs となっていて、phrd1 は d 節、第 1 章のフレーズを意味する。

●縁を付けた演算表

```
   bft
+-+-+--------------------+
|1|:|i. by i. over x./~@i.|
+-+-+--------------------+
   * bft 4
+-+-------+
| |0 1 2 3|
+-+-------+
|0|0 0 0 0|
|1|0 1 2 3|
|2|0 2 4 6|
|3|0 3 6 9|
+-+-------+
```

後にも出てくるが、縁取りされた関数表を作る動詞である。
この例は 0 から 3 までの足し算表である。

●ロケール

単に読み込んだだけでは重複した名前は後のものが優先されてしまう。
そこで左引数にロケールの名前を付けて読み込むとよい。
使用するときはロケール名の前後に_を付けて名前の後ろに付け加える。

```
   'a2'load 'system\examples\phrases\phra2.ijs'
   m67_a2_
+-+-+--+
|1|&|o.|
+-+-+--+
   m67_a2_ 0 1
0 0.841471
   m67
|value error: m67
```

ロケール名を付けないと認識されない。

```
   sin_a2_ 0 1
0 0.841471
```

●読み込まれた関数の一覧表

```
   y=:1!:1<'system\examples\phrases\phra1.ijs'
```

```
   y
&  Phrases from Chapter 1, Section A of J Phrases.

n0=: i. 6                        NB.List of first six non-negative integers
m1=: ^&3                         NB.Cube
m2=: mean=: +/ % #               NB.Arithmetic mean
d3=: $,:                         NB.x copies of y
m4=: <.@(0.5&+)                  NB.Round
m5=: =+                          NB.Test for real number
m6=: (<0 _1)&C.                  NB.Swap leading and final items
m7=: +/\  -: +/\.&.|.            NB.Prefix sum scan is suffix under reversal
```
ファイルを読み込む動詞 1!:1 を使って y に入れ、プリントすればよい。

C 節． 乱数表の作成

引数としてアレーを使用するとき、i.2 3 や i.3 2 4 のようなシステマティックな数値を用いると、真のテストにならない場合がある。そのとき使われるのが乱数である。ただし、通常の乱数では再びテストしたときその度に答えが違っていては煩わしい。そこで使われるのが seed に影響されない乱数を作成動詞?. である。

なお、?. はランダムシードとして $7^5 = 16807$ を使っている。すなわち?を使用するとき、直前に 9!:1]16807 を実行することと同じである。

d0=: QI=: ?.@$	i.y からの形状 x の整数乱数
d1=: QD=: ?.@(,:~)@$	i.y からの形状 (2,x) の整数乱数
d2=: QN=: -/@QD	正と負の整数乱数
d3=: QF=: %/@QD	正の有理数乱数
d4=: QC=: j./@QD	正の複素数乱数
d5=: QA=: {&a.@(a.&i.@] ＋ [QI 26"_)	y で始まる 26 文字からなる文字乱数
m6=: QI&10	一桁整数からなる乱数
m7=: QZ=: QI&1	ゼロからなる形状 x のアレー
m8=: QB=: QI&2	ゼロと 1 からなる (Boolean) 形状 x のアレー

●乱数アレー

乱数アレーは？2 3 4$10 を使えばつくることができるが、繰り返し実験には不向きである。
基本的には QI を用いるのであるが、これを使えば特定の seed で乱数をつくることができる。

```
   (QA&'A';QA&'a';QA&'+')2 3 4
+----+----+----+
|DTLN|dtln|.>68|
|FBRR|fbrr|0,<<|
|YJNV|yjnv|C48@|
|    |    |    |
|ABNR|abnr|+,8<|
|AJBK|ajbk|+4,5|
|RPYW|rpyw|<:CA|
+----+----+----+
```

常に同じ数で始まる乱数が使用される。

●ボックス化
次のフレーズは引数をいろいろなセル単位でボックス化したり、スケール変換や正規化などを行うときに用いられる。

a9=: B=: <"	副詞：k B はランク k のボックスをつくる
m10=: B0=: 0 B	アトムのボックスをつくる
m11=: B1=: 1 B	リストのボックスをつくる
m12=: B2=: 2 B	テーブルのボックスをつくる
m13=: RG=: (1: ＋ >./ - <./)@,	y の範囲、max-min+1。何故+1？
d14=: SC=: * % RG@]	y の範囲を x にする
m15=: NR=: 1&SC	範囲を 1 にする
m16=: SZ=: - <./@,	最小値が 0 になるようにずらす
m17=: NM=: SZ@NR	範囲が [1， 0] になるように正規化する

B0、B1 でランクを指定することによりいろいろなボックス化ができる。

```
   B0 B1 2 3 4 QA 'A'
+------+------+------+
|+----+|+----+|+----+|
||DTLN|||FBRR|||YJNV||
|+----+|+----+|+----+|
+------+------+------+
|+----+|+----+|+----+|
||ABNR|||AJBK|||RPYW||
|+----+|+----+|+----+|
+------+------+------+
   B2 3 4 5 QA 'a'
+-----+-----+-----+
|dtlnf|rpywn|zstqb|
|brryj|crksx|qxhlt|
|nvabn|tgbti|mghje|
|rajbk|qtzjg|mxxbx|
+-----+-----+-----+
```

3 × 4 の乱数行列についていろいろな変換を行う。

```
      (];NR;SZ;NM)3 4 QN 10
+----------+------------------------------+---------+--------------------------+
|1  7 _1 _1|0.0833  0.5833 _0.0833 _0.0833|5 11  3 3|0.4167 0.9167   0.25   0.25|
|2 _3  6  2|0.1667    0.25     0.5  0.1667|6  1 10 6|   0.5 0.0833 0.8333    0.5|
|3 _2 _4  0|  0.25 _0.1667 _0.3333       0|7  2  0 4|0.5833 0.1667      0 0.3333|
+----------+------------------------------+---------+--------------------------+
   ]            スケール幅を 1 とする          最小値を 0        範囲を [0,1] とする
```

D 節． 演算表

和、差、積などの演算表を作成するときに便利な動詞や副詞などを掲げる。

d0=:＋/	和の演算表
d1=:*/	積の演算表
d2=:>./	最大値の表
d3=:[by] over ＋/	縁付き、和の演算ん表
d4=:by=:’ ’&;@,.@[,.]	横に縁を付ける動詞
d5=:over=:({.;}.)@":@,	上に縁を付ける動詞
m6=:＋/~@i.	始めの y 個の整数の和の演算表
m7=:bc=:!/~@i.	y次までの 2 項係数表
a8=:ft=:(/ ~) (@i.)	始めの y 個の整数について、演算表をつくる副詞
a9=:bft=:1 : ’i.by i.over x./~@i.’	縁を付けて、y 個の整数について、演算表をつくる副詞

```
   2 3 5(d0;d1;d2)0 1 2 3 4 5
+-----------+--------------+-----------+
|2 3 4 5 6  7|0 2  4  6  8 10|2 2 2 3 4 5|
|3 4 5 6 7  8|0 3  6  9 12 15|3 3 3 3 4 5|
|5 6 7 8 9 10|0 5 10 15 20 25|5 5 5 5 5 5|
+-----------+--------------+-----------+
```

和の演算表　　積の演算表　　最大値の演算表

```
   2 3 5 d3 0 1 2 3 4 5
+-+------------+
| |0 1 2 3 4  5|
+-+------------+
|2|2 3 4 5 6  7|
|3|3 4 5 6 7  8|
|5|5 6 7 8 9 10|
+-+------------+
```

縁をつけた和の演算表

```
   (m6;m7)5
+---------+---------+
|0 1 2 3 4|1 1 1 1 1|
|1 2 3 4 5|0 1 2 3 4|
|2 3 4 5 6|0 0 1 3 6|
|3 4 5 6 7|0 0 0 1 4|
|4 5 6 7 8|0 0 0 0 1|
+---------+---------+
```

0〜5 の和の演算表と 4 次までの 2 項係数表

```
   % a9 5
+-+--------------------+
| |0 1   2        3    4|
```

```
+-+--------------------+
|0|0 0   0        0    0|
|1|_ 1 0.5 0.333333 0.25|
|2|_ 2   1 0.666667  0.5|
|3|_ 3 1.5        1 0.75|
|4|_ 4   2  1.33333    1|
+-+--------------------+
```

縁を付けた 0〜5 の割り算の演算表

２００２年2月２３日

Jのフレーズについて
＿＿＿＿　第２回　＿＿＿＿

慶応義塾大学理工学部
竹内寿一郎

第２章　原始関数にもとづくフレーズ

A 節．　ボンド (&) および、ちょっとした副詞

簡単ではあるが使って便利な、原始関数を使ったフレーズを掲げておく。

m0=: 1&＋	1 を加える　¿:
m1=: ＋&1	〃
m2=: _1&＋	1 を引く　¡:
m3=: -&1	〃
m4=: 1&-	否定　-. (論理否定、または 1 から引く)
m5=: 1&~:	論理否定
m6=: 0&=	〃
m7=: 0&-	符号を変える
m8=: _1&*	〃
m9=: *&_1	〃
m10=: 2&*	2 倍　＋:
m11=: *&2	〃
m12=: 3&*	3 倍
m13=: *&3	〃
m14=: 0j1&*	j.　虚数を乗ずる
m15=: ^@j.	r.　y ラジアンの単位円周上の複素数
m16=: 1p1&*	π 倍　o.
m17=: 0.5&*	2 分の 1　-:
m18=: *&0.5	〃
m19=: %&2	〃
m20=: 1&%	逆数　%
m21=: ^&_1	〃
m22=: ^&2	2 乗 *:
m23=: ^&3	3 乗
m24=: ^&0.5	平方根 %:
m25=: ^&1r2	〃
m26=: 2&%:	〃
m27=: ^&(%3)	3 乗根
m28=: ^&1r3	〃
m29=: 3&%:	〃
m30=: (^1)&^	指数関数 ^
m31=: 1x1&^	〃

m32=: 1x1&^.	自然対数 ^.
m33=: 10&^	10 のべき乗
m34=: 10&^.	常用対数
m35=: >:@<.@(10&^.)@(1&>.)	整数 y を表すのに必要な桁数
m36=: #@(10&#.^:_1)"0	〃
m37=: >:@<.@(2&^.)@(1&>.)	整数 y を表すのに必要なビット数
m38=: #@(2&#.^:_1)"0	〃
m39=: 0&{	先頭のアイテム {.
m40=: _1&{	最後尾のアイテム
m41=: 1&}.	先頭のアイテムを落とす }.
m42=: _1&}.	最後尾のアイテムを落とす }:
m43=: 0&<	正の値のテスト
m44=: 0&>	負の値のテスト
m45=: 0&>.	Max (0,y)
m46=: 0&<.	Min (0,y)
m47=: (0&=)@(2&|)	偶数のテスト
m48=: (1&=)@(2&|)	奇数のテスト
m49=: _1&A.	逆順 —.
m50=: (<0 _1)&C.	先頭と最後尾のアイテム交換
m51=: <.@(0.5&+)	まるめ
m52=: ,~ $ 1: ,] $ 0:	y 次の単位行列
m53=: -.@(' '&E.) #]	2 個以上のブランクを除く
m54=: BC=: i.@>: !]	y 次の 2 項係数
m55=: (0&, +,&0)^:(['1:)	〃 (再帰的)
m56=: BCT=:i. !/ i.	y-1 次までの 2 項係数表
m57=: PAT=: |:@BCT	パスカルの三角形
m58=: (0&, +,&0)^:(i.'1:)	〃 (再帰的)
m59=: IX=: a.&i.	アルファベットのアスキー指標化
m60=: Lt=:(1&e.)@(e.&a.)@,	文字列テスト
m61=: 1&#.	最終軸に関する和 (リストの要素和)、 +/"1
m62=: 1&,	先頭に 1 からなるアイテムを連結
m63=: ,&1	最後尾に 1 からなるアイテムを連結
m64=: 1&,.	アイテムの先頭に 1 からなるアイテムのアイテムを連結
m65=: ,.&1	アイテムの最後尾に 1 からなるアイテムのアイテムを連結
m66=: 1&,@$ $,	全体をアイテム化する (形状の先頭に 1 を付加する) ,:

●上の表で分かりにくい動詞についていくつか説明をする。

```
   m35=: >:@<.@(10&^.)@(1&>.)        整数 y を表すのに必要な桁数
   m36=: #@(10&#.^:_1)"0             整数 y を表すのに必要な桁数
   m35 23456.78
5
   m36 _23456.78
5
```

y は正の数とする。m35 はまず y が負または 1 より小さいときを排除するため 1&>. で 1 以上の数にする。

10&^. で常用対数をとり、切り捨ててから 1 を加えると必要な桁数を得る。
m36 は 10 10 10&#: 123 を用いると 10 10 10 が必要となるので、10&#.1 2 3 が 123 になることを利用、
その逆関数を上手に使っている。10&#.^:_1[_123 は 8 7 7 と補数が出るので、負の数でもよい。

```
m37=: >:@<.@( 2&^.)@(1&>.)          整数 y を表すのに必要なビット数
m38=:#@( 2&#.^:_1)"0                整数 y を表すのに必要なビット数
   m37 256
9
   m38 _256
9
```
m35、m36 において 10 の代わりに 2 を用いている。

```
   m52=: ,~ $ 1: , ] $ 0:           y 次の単位行列
   m52 3
1 0 0
0 1 0
0 0 1
```
$の右のフォークにより 1 0 0 0 ができ、,~で (3,3) ができ、3 3$1 0 0 0 で単位行列ができる。

```
   m53=: -.@(’  ’&E.) # ]           2 個以上のブランクを除く
   m53 ’I   love you    .’
I love you .
   ’  ’&E. ’I   love you    .’
0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0
```
2 つ以上のブランクが全て 1 となる。

```
   -.0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0
1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1
```
not(-.) をとることにより、2 つ以上のブランクを除くべきリストを得る。

```
   m55=: (0&, ＋,&0)^:([ ‘1:)       2 項係数（再帰的）
   m55 4
1 4 6 4 1
   (0&,+,&0)^:4[1
1 4 6 4 1
   (0&,+,&0)^:(]‘1:)4
1 4 6 4 1
```
(0&,+,&0) の引数として ([‘1:) の動詞 (1:)4 からできる 1 が採用される。
その結果を何回か繰り返すと 2 項係数を得ることができる。次の m58 を参照のこと。

```
   m58=: (0&, ＋,&0)^:(i.‘1:)        パスカルの三角形（再帰的）
   m58 4
1 0 0 0
```

```
1 1 0 0
1 2 1 0
1 3 3 1
   (0&,+,&0)^:0 1 2 3 4[1
1 0 0 0 0
1 1 0 0 0
1 2 1 0 0
1 3 3 1 0
1 4 6 4 1
```
繰り返しのところをリストにすることができる。

```
   m60=: Lt=:(1&e.)@(e.&a.)@,      文字列テスト
   m60 ?3 3$10
0
   m60 ":?3 3$10
1
```
e.&a.'123' は 1 1 1、1&e.1 0 0 1 は 1。リスト化して、一つでも 1 があれば文字列と見なす。

```
   m66=: 1&,@$$,                   全体をアイテム化する（形状の先頭に 1 を付加する） ,:に同じ。
   m66 ?3 3$10
0 4 6
5 9 8
5 0 6
   $m66 ?3 3$10
1 3 3
```
m66 を実行すると、見かけは同じであるが形状は元の形状よりランクがあがり、先頭に 1 が加えられる。
(1&,@$) と$と , のフォークからなっていて、 , でリスト化された y. を (1,$y.) の形状に並べ替える。

●以下は円関数のリストである。このように定義しておくと使い易い。

m67=: sin=: 1&o.	正弦関数 (Sin)
m68=: asin=: _1&o.	逆正弦関数 (Arcsin)
m69=: cos=: 2&o.	余弦関数 (Cos)
m70=: acos=: _2&o.	逆余弦関数 (Arccos)
m71=: tan=: 3&o.	正接関数 (Tan)
m72=: atan=: _3&o.	逆正接関数 (Arctan)
m73=: sinh=: 5&o.	双曲線正弦関数 (Sinh)
m74=: asinh=: _5&o.	逆双曲線正弦関数 (Arcsinh)
m75=: cosh=: 6&o.	双曲線余弦関数 (Cosh)
m76=: acosh=: _6&o.	逆双曲線余弦関数 (Arccosh)
m77=: tanh=: 7&o.	双曲線正接関数 (Tanh)
m78=: atanh=: _7&o.	逆双曲線正接関数 (Arctanh)

●次に便利なちょっとした副詞を掲げる。

`a79=: each=: &.>`	左引数の関数を各ボックスに適用する
`a80=: inv=: ^:_1`	逆関数
`a81=: ^:_`	極限
`a82=: (D.1) " 0`	1次微分
`a83=: ;.1`	分割
`a84=: !.0`	正確な比較 (精度一杯比較する)；例えば = a84
`a85=: "0`	アトムの定数関数
`a86=: "1`	リストの定数関数
`a87=: "_1`	アイテムの定数関数
`a88=: "_`	任意の型の定数関数
`a89=: 0!:`	スクリプト；表示なしでスクリプトを読むには se=: 0 a89
`a90=: file=: 1!:`	ファイルを読む；例えば read=:1 file

２００２年3月１６日

Jのフレーズについて
＿＿＿＿ 第３回 ＿＿＿＿

慶応義塾大学理工学部
竹内寿一郎

第２章　原始関数にもとづくフレーズ（続き１）

B 節．　曖昧項動詞

曖昧項動詞 h は`h=:f : g`で定義され、f が単項動詞、g が 2 項動詞で定義しなければならない。これらの動詞は予め定義されていれば明示的 (Explicit) であっても黙示的 (Tacit) であってもよい。また単項、2 項のいずれか 1 つが定義されているとき、他方の定義にあたり、自己参照 (Self-Reference) を用いた黙示的定義であらわしてもよい。

`v0=: 10&^. : ^.`	対数、単項のときは常用対数
`v1=: 10&$: : ^.`	上と同様、ただし単項のとき自己参照を使用
`v2=: 10&^. :($:*^.@(10"0)%^.@[)`	上と同様、ただし 2 項のとき自己参照を使用
`d3=: res=: [: : |`	単項は空 (2 項動詞のみ)
`m4=: abs=: | : [:`	2 項は空 (単項動詞のみ)

●原本にちょっとしたミスがありました。

```
   v2=.10&^. :($: * ^.@(10"0) % ^.@[)
   v2_1=.10&^. :($:@] * ^.@(10"0) % ^.@[)
   2 v2 64
|limit error: v2
|   2     v2 64
   2 v2_1 64
6
```

v2 は`$:`と`*`と`(^.@(10"0)%^.@[)`のフォークで、この括弧の中もフォークという構成である。したがって本来単項のはずの`$:`が 2 項で働いてしまって、自己参照を無限回繰り返すという矛盾を生じている。それゆえ v2_1 のように`$:`のところを`$:@]`のように単項として働くべく修正しなければならない。
10 から 2 への対数の底の変換は次の式による。

```
   (10^.64)*(^.10)%(^.2)
6
```

●関数定義で、単項または 2 項の動詞を無効にするには`[:`を空に働かせればよい。
d3 および m4 はその例である。動詞を黙示的に作成した場合、本人の意思を無視して異なった働きをするときがあるので、それを防ぐには明示的に定義して、単項または 2 項のときにのみ働くように制限しておくとよい。この定義方法は参考になる。

C 節．　副詞と接続詞

接続詞からなる副詞

接続詞は 1 つの引数をとると副詞を構成する。たとえばa1=:&3 ならばˆa1 はˆ&3 すなわち 3 乗という関数になる。もしもa2=:3&であればˆa2 は 3 のべき乗という関数になる。接続詞つくられた副詞は数多くあり、その種類も多岐にわたっている。それらの副詞を次に掲げる。

a0=: I=: ˆ:_1	逆関数 (ˆI はˆ.)
a1=: L=: ˆ:_	極限 (2&o.L 1 は y=cos y の解)
a2=: LI=: ˆ:__	Limit of inverse
a3=: SQ=: ˆ:2	2 回繰り返し (1&o.SQ は正弦を 2 回繰り返す)
a4=: C=: &o.	円関数の定義 (3 C は正接関数)
a5=: CO=: %@C	3 CO は正接関数の逆数
m6=: rfd=: 1r180p1&*	角度からラジアンへ
m7=: dfr=: rfd I	ラジアンから角度dfr=: dfr f. は定義を固定する
a8=: D=: @rfd	角度からラジアンに変換して ··· 。試してみよ 1 C D 0 30 45 60 90 180
m9=: SIN=: 1&o. D	角度を引数とする正弦関数
a10=: T=: "2	テーブル単位のボックス化。試してみよ。<T i. 2 3 4 3
a11=: S=: ˆ!.	一般化した階乗 (Stope factorial)
a12=: P=: p.!.	多項式の Stope
a13=: FILL=: |.!.	埋め込みながらシフトする。周期的にシフトしない
a14=: FILE=: 1!:	ファイル関数 (1 FILE は読み込み、等)

●接続詞!. について

接続詞!. は比較や切り捨てなどの動詞に働いて精度を設定する。例えば

```
   2<:1.999999999999
0
   2<:1.9999999999999
1
   2<:!.(1e_11)1.9999999999
0
   2<:!.(1e_11)1.99999999999
1
   2<:!.(1e_12)1.99999999999
0
   2<:!.(1e_12)1.999999999999
1
   2<:!.(1e_19)1.999999999999
0
   2<:!.(1e_19)1.9999999999999999
1
   2<:!.(1e_10)1.9999999999999999
|domain error
|   2    <:!.(1e_10)2
```

精度は 1e_11 より小さい値でなければならず、1e_18 ないしは 1e_19 より小さくしても、コンピュータの精度 (倍精度計算) 以上にはならない。

副詞 a11=:S=:^!. は特殊な働きをする。例えば

```
   3 ^!.2 [4
945
```

は */3+2*i.4 なる計算を行う。J の Dictionary の中の!. の説明に x ^!. r n は */x+r*i.n と書いてある。

```
   */3+2*i.4
945
```

★★★ところで、副詞と副詞句？の違いに注意しよう。★★★

```
   3 (^!. 2) 4
945
   3 (2 a11) 4       注意：3 (all 2) 4 ではエラーとなる。
945
   3 (2 S) 4
945
   3 (2 (^!.)) 4     注意：3 ((^!.) 2) 4 ではエラーになる。
945
```

以上これらの違いを考えてみよ !!
そうなのである。^!. は副詞句？で引数は右にとる。
(^!.) はれきっとした副詞なので引数は左にとる。
勿論 a11=:S=:^!. だから a11 も S も副詞である。
このことは vvv は動詞が 3 つならんだもの、(vvv) はフォークであることからも想像ができよう。

●許容誤差
さて、辞書でみると!. を伴った許容誤差に関する動詞には
< <: > >: +. *. -. -: | E. i. i: <. >. * ~. = ~: #: e. 等がある。
例えば=!. について調べてみる。

```
   1r3=!.1e_11[0.333333333333
1
   1r3=!.1e_11[0.33333333333
0
   1r3=!.1e_12[0.333333333333
1
   1r3=!.1e_13[0.333333333333       精度が働いて等しくならない !!
0
   1r3=!.1e_10[0.333333333333
|domain error
|   1r3    =!.1e_10[0.333333

   i.4.99999999999999999999
0 1 2 3 4
   i.4.9999999999999
0 1 2 3 4
```

```
   i.4.999999999999
|domain error
|       i.5
   i.!.1e_11[4.9999999999999
0 1 2 3 4
   i.!.1e_11[4.999999999999        桁数が不足でエラーとなる
|domain error
|       i.!.1e_11[5
```

精度は 1e_11 以下でなければ domain error となる。=!以外では結果は精度に関係なく？演算される。
これらについて幾つか調べてみたが、APL のようにきちんとした許容誤差による結果を得ることが出来なかった。システムのバグではないかと思われる。

許容誤差以外では^!.(Stope Factorial)、p.!.(Stope Polynomial)、$ |. , ,. ,: # {. (Fill)、":(Print F
などがある。その中で"Stope"は何の目的で、計算ルールを決めているのかよく分からなかった。

● a11=:S=:^!. は一般化した階乗ということができる。(x ^!. r n は */x+r*i.n)

$$\Pi_{i=0}^{n}(r*i+x)$$

x=1、r=1 ならば y の階乗、r=0 ならば x^y である。

```
   1 (1 all) 6                  x=1,r=1 のときは階乗
720
   !6
720
   5 (0 all) 4                  r=0 のときは 5^4
625
   5^4
625
   1 (2 all) 4                  x=1,r=2 のときは 1*3*5*7
105
   */1 3 5 7
105
   2 (2 S) 4                    x=2,r=2 のときは 2*4*6*8
384
   */2 4 6 8
384
```

a12=: P=: p.!. の使い方はさっぱり見当がつかなかった。

```
   1 3 3 1(p.!.1)0 1 2 3
1 16 49 106
   1 3 3 1(p.!.2)0 1 2 3
1 28 79 160
   1 3 3 1(p.!.3)0 1 2 3
1 44 117 226
```

●Fill

```
   2 4($!. 10)1 2 3 4 5
1  2  3  4
5 10 10 10
   (2 4)'T'($!.)'nishi'
nish
iTTT
   10{.1 2 3 4 5
1 2 3 4 5 0 0 0 0 0
   10({.!.10)1 2 3 4 5
1 2 3 4 5 10 10 10 10 10
   10'T'({.!.)'nishi'
nishiTTTTT
```

この例からも分かるように、不足分を指定された数または文字で埋めている。

```
   3 (|.!.10)1 2 3 4 5
4 5 10 10 10
   3 'T'(|.!.)'nishikawa'
hikawaTTT
```

これも 3 個のシフトで、シフトした分が 10 または'T' で埋められている。

残りの , ,. ,: # については役割が確かめられなかった。

●プリント精度

```
   ":!. 5[0.11111111111111111111111111111111
0.11111
   ":!. 8[0.11111111111111111111111111111111
0.11111111
   ":!. 8[1111.11111111111111111111111111111
1111.1111
   ":!. 16[0.11111111111111111111111111111111
0.1111111111111111
   ":!. 17[0.11111111111111111111111111111111
0.1111111111111111
   17 (":!.)0.11111111111111111111111111111111
0.1111111111111111
```

最大でも 16 桁しか印刷してくれない。

明示的定義

J の初心者にとって、副詞や接続詞の関数を読んだり書いたりするときは明示的の方が扱いやすい。そこで関数定義のときの 1 :や 2 :の代わりに 11 :または 12 :を用いると、黙示的定義を得ることができる。

```
   split=:2 :',.@(x.@(y.&{.);x.@(y.&}.))'
```

```
   (+:split 2,.|.split 3,.+/split 2)i.5 3
+--------+--------+--------+
|0 2  4  |6 7 8   |3 5 7   |
|6 8 10  |3 4 5   |        |
|        |0 1 2   |        |
+--------+--------+--------+
|12 14 16|12 13 14|27 30 33|
|18 20 22| 9 10 11|        |
|24 26 28|        |        |
+--------+--------+--------+
   tacitsplit=:12 :',.@(x.@(y.&{.);x.@(y.&}.))'
   split
+-+-+-------------------------+
|2|:|,.@(x.@(y.&{.);x.@(y.&}.))|
+-+-+-------------------------+
   9!:3(5)
   tacitsplit
,. @ (([. @ (]. & {.)) ; ([. @ (]. & }.)))
```

黙示的定義では表示すると、自動的にいくつかのスペースが入るが、このスペースはなくてもよい。

次の例では黙示的動詞d16,d17 を利用して明示的に副詞a18 を作成している。

`c15=: ,.@([.@(].&{.);([.@(].&}.)))`	split の黙示的定義
`d16=: by=: ' '&;@,.@[,.]`	表の見出しを横に付けるための動詞
`d17=: over=: ({.;}.)@":@,`	表の見出しを上に付けるための動詞
`a18=: tab=: 1 :'[ by ]over x./'`	左と上に見出しが付いた関数表を作成するための副詞

掛け算表

```
   1 2 3*a18 1 2 3 4
+-+--------+
| |1 2 3  4|
+-+--------+
|1|1 2 3  4|
|2|2 4 6  8|
|3|3 6 9 12|
+-+--------+
```

べき乗表

```
   1 2 3^a18 1 2 3 4
+-+---------+
| |1 2  3  4|
+-+---------+
|1|1 1  1  1|
|2|2 4  8 16|
|3|3 9 27 81|
+-+---------+
```

名詞の引数

副詞が名詞の引数をとったり、接続詞が左右に一つの名詞と一つの動詞をとるケースはよく現われる。

```
   x=:0 0 1 1[y=:0 1 0 1
   x*.y                          論理積
0 0 0 1
   x 1 b. y                      副詞 b. を使った論理積
0 0 0 1
   x(i.16)b.y                    副詞 b. を使った全ての論理演算
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
   C=:&o.                        円関数の副詞
   1 C 0 1r4p1 1r3p1 1r2p1 1p1    1 が引数のときは正弦関数
0 0.707107 0.866025 1 0
   ^&3 i.6                       接続詞&が左右に引数をとる
0 1 8 27 64 125
   2|.!. 1 i.6                   接続詞!. が左右に引数をとる
2 3 4 5 1 1
```

●接続詞が左右に名詞の引数をとって動詞をつくるケースは稀である。

```
   a=:1 4 6 4 1[b=:1 2 1
   RAT=:[.&p.%[.&p.)             多項式の割り算を定義する接続詞
   4!:0< 'RAT'                   接続詞であることを確かめる
2
   9!:3(5)                       Linear な表現形式にする
   a RAT b                       多項式 a p.z を b p.z で割り算をする動詞
1 4 6 4 1&p. % 1 2 1&p.
   a RAT b z=:i.6                 z=0 1 2 3 4 5 での割り算の結果
1 4 9 16 25 36
   b RAT a z                     逆に多項式 b p.z を a p.z で割った結果
1 0.25 0.111111 0.0625 0.04 0.0277778
   (a RAT b*b RAT a)z             割り算を入れ替え、積をとると 1 になる
1 1 1 1 1 1
```

●多項式の擬表示　数学では $2x^3 + 4x^2$ というような表現がなされるが、J においても2 x 3 ＋ 4 x 2 のように書けたら便利で読みやすくなる。

```
   x=:[.&*@(^&].)"0               まず、接続詞を定義する
   2 x 3                         左右に名詞の引数を付けて 2*x^3 を定義する
2&*@(^&3)"0
   2 x 3 z=:i.6                   実際に z=0 1 2 3 4 5 について計算すると
0 2 16 54 128 250
   2*z^3                         確かに 2*z^3 になっている
```

```
0 2 16 54 128 250
   2 3 5 x 1 2 4 z              2*z 3*z^2 5*z^4 を同時に計算してみる
 0  0    0
 2  3    5
 4 12   80
 6 27  405
 8 48 1280
10 75 3125
```

そこで新たに x を定義しなおす。

```
   x=:+/@([.&*@(^&].))"0        和を追加して x を再定義する
   2 3 5 x 1 2 4 z              2x+3x^2+5x^4 の計算ができる
0 10 96 438 1336 3210
   (2 x 1 + 3 x 2 + 5 x 4)z      x は接続詞であるからこのように書ける
0 10 96 438 1336 3210           +の前後にスペースは要らないが、見やすくした
   2 x 3 z                      勿論 1 つの項だけでもよい
0 2 16 54 128 250
```

c19=: RAT=: [. & p. % (]. & p.)	有理多項式を作る接続詞
c20=: x=: ＋/@([.&* @ (^&].))"0	初等数学のための表示用接続詞
c21=: bind=: [.@("_)	単項関数 x. と y. との連結

●関数のバインド

最後に関数などをつないで再定義するための接続詞c21 について述べる。

```
   f1=:*:                       第 1 関数を定義する
   f2=:3&*                      第 2 関数を定義する
   f2 f1 5                      連続関数として計算する
75
   f3=:f2 c21 f1                接続詞 c21 を使って連結し、定義し直す
   f3 5                         新しい単項関数で計算する
75
   f31=:[:f2 f1                 単に連続させるとフックになるので [:で分離する
   f31 5                        この形はキャップドフォークと呼ばれる
75
   wdinfo 'Job Finished'        ダイアログボックスが開いて Job Finished と聞いてくる
   fini=:wdinfo c21 'Job Finished'
   fini''                       引数なしで同じ結果を得る

   fini=:wdinfo 'Job Finished'  ダイアログボックスが開いて Job Finished と聞いてくる
   fini                        fini には何も入っていない
   fini ''                      実行するとエラーになる
|syntax error
|       fini''
```

２００２年4月２７日

Jのフレーズについて
＿＿＿＿ 第４回 ＿＿＿＿

慶応義塾大学理工学部
竹内寿一郎

第２章　原始関数にもとづくフレーズ（続き２）

D 節．　明示的定義

明示的定義は読み書きに便利であるだけでなく、if then else が使用できるので、他の言語によるプログラミングと違和感なく使うことが出来る。また 1 行で書ける文は容易に黙示的定義へ変換することが出来るという特長もある。

● 13 : 0 で動詞を 1 行で定義すると、黙示的定義の動詞を得る。

```
   log=:3 : '10^.y.'            常用対数の明示的定義
   log 1 10 20 40 100
0 1 1.30103 1.60206 2
   9!:3(5)                     Linear 形式の表示を促す
   log
3 : '10^.y.'
   log=:13 : '10^.y.'          13 : で常用対数の動詞を定義してみる
   log 1 10 20 40 100
0 1 1.30103 1.60206 2
   log
10"_ ^. ]                      常用対数の黙示的定義を得る
```

a0=: def=: : 0	明示的定義をするための副詞

●関数の明示的定義

1 : 0 , 2 : 0 , 3 : 0 は副詞、接続詞、動詞の明示的定義に用いられる慣用句であり、def を用いると引用符 ' を使わずに関数を定義することが出来る。

```
   def=: :0                副詞を定義 (この 0 は以下の文を読み込むことを意味する)
   4!:0<'def'              品詞を調べると、副詞であることが分かる
1
   LOG=:3 def              def を使って LOG を定義する
10&^.y.                     単項で使うと常用対数
:
x.^.y.                      2 項で使うと、x を底とする対数
)
   LOG 1 10 20 40 100
0 1 1.30103 1.60206 2
   10 LOG 1 10 20 40 100
0 1 1.30103 1.60206 2
```

```
   rat=:2 def                    有理多項式の計算のための接続詞を定義
x.&p.%y.&p.                       x. y. はそれぞれ多項式の係数
)
   1 4 6 4 1 rat 1 2 1           (x+1)^4 を (x+1)^2 でわる動詞
1 4 6 4 1&p. % 1 2 1&p.
   (1 4 6 4 1 rat 1 2 1)i.6      x に 0 1 2 3 4 5 を入れた有理多項式の計算
1 4 9 16 25 36
```

●行列の入力

;._2 と (0 : 0) を利用すると行列を直接入力することが出来る。0 : 0 は入力によって名詞を定義、;._2 は最後尾の CR(または LF かも知れない) が fret となって、'oneCRtwoCRthreeCR' という文を CR で区切っている。

```
   mat=:[;._2(0 : 0)      入力による名詞 mat の定義
one                       この最後尾は CR
two                       最後尾は CR
three                     この最後尾の CR が fret となっている
)
   mat
one
two
three
   $mat
3 5
   matrix=:0".];._2(0 : 0)      0". で数値化すると数値行列が入力できる
1 2 3 4
2 1 5 6
3 5 1 7
4 6 7 1
)
   $matrix                    文字ではなく数値であることが分かる
4 4
   matrix+1                    従って計算が可能である
2 3 4 5
3 2 6 7
4 6 2 8
5 7 8 2
   boxed=:<;._2(0 :0)         ] の代わりに<を使うとボックスで囲まれる
one
two
three
)
   boxed
+---+---+-----+
```

```
|one|two|three|
+---+---+-----+
```
なお、(0 : 0) を使って行列を定義するとき、return キィを押してしまった行は修正することが出来ない。

●動詞にランクを付与することができる。
```
   fn=:3 : 0"1
<+/y.
)
   fn i.2 3 4
+--+--+--+
|6 |22|38|
+--+--+--+
|54|70|86|
+--+--+--+
```

●関数名（動詞の名前）を与えない関数の実行。
```
   x=:3
   3 : 0 x
if.2|y.do.'odd'else.'even'end.
)
odd
```
これは if.do.else. などのアルゴリズムの簡単なテストに使えそうである。

E 節.　フック

括弧で括られた 2 つの動詞はフックを構成する。単項のフック(uv)y はy u v y なる働きをする。また、2 項フックx (uv)y はx u v y なる働きをする。なお、フックではu は 2 項動詞、v は単項動詞でなければならない。

m0=: It=: =<.	整数であるかどうかのテスト
m1=: Rt=: =+	実数であるかどうかのテスト
v2=: $,:	y を形状 x でコピー
m3=: cf=: (+%)/	連分数の計算
m4=: cfc=: (+%)/\	連分数を逐次表示しながら計算

フックはこの本のいたるところで出てくるし、フォークと並んで J 言語の大きな特長の一つである。

●整数および実数テスト
```
   (=<.)y=:1 1.5 _2 _2.5 2j2 1.5j2        まず、整数テストを試みる
1 0 1 0 1 0
   y=<.y                                  y は y を切り捨てた値、即ち y の整数部に等しい
1 0 1 0 1 0
   Rt 2.555 1j0.00005 1j1e_13 1j1e_14 0.0
1 0 0 1 1                                 実数値テスト。1e_14 はゼロと見なされる
```
●連分数

```
   cf 3 7 5 1                          連分数による円周率の近似
3.13953
   (+%)/3 7 5 1                        (+%) を挿入して計算
3.13953
   3(+%)7(+%)5(+%)1                    2項フックが次々に働いて計算
3.13953
   3+%(7+%(5+%1))                      連分数の定義による計算
3.13953
   cfc 3 7 5 1                         収束の途中経過を示す
3 3.14286 3.13889 3.13953
   cfc 1 1 1 1 1 1 1                   連分数による黄金比の計算
1 2 1.5 1.66667 1.6 1.625 1.61538
```

F 節． 動詞のトレイン

動詞が 3 つ以上並んで括弧で括られていると動詞のトレインと呼ばれる。このとき右から順に 3 個の動詞でフォークを構成し、これが 1 つの動詞と解釈される。次にこの動詞も含め右から 3 個の動詞が纏まって 1 つのフォークとなり、順次 3 つずつ右から 3 個ずつフォークが構成される。このとき、最後に動詞が 2 個しかなければフックとなって終了する。動詞のトレインは最後はフォークかフックで終了する。

フォークの機能を解除するときは動詞の間に@を入れるか、キャップ[:を使用する。この 2 つの方法は全く同じではないので十分注意して使い分けねばならない。

`m0=: >: @ ＋: @ i.`	始めの幾つかの奇数
`m1=: 1: ＋ 2: * i.`	m0 と同じ
`m2=: ＋/ @ (1: ＋ 2: * i.)`	1 で始まる奇数の和
`m3=: [: ＋/ 1: ＋ 2: * i.`	m2 と同じ、キャップ [:を使った表現
`v4=: m=: [:`	m はキャップであると定義する
`m5=: m ＋/ 1: ＋ 2: * i.`	m3 と同じ
`m6=: m2 -: *:`	m2 は奇数の 2 乗和に等しい (Tautology)
`m7=: %:@(＋/@(*:@(]-＋/%#)))`	平方和の平方根
`m8=: m%: m ＋/ m*: ] - ＋/ % #`	m7 と同じ

```
   (>:@+:@i.)5                         単項動詞を次々に実行
1 3 5 7 9
   (1:+2:*i.)5                         動詞のトレイン
1 3 5 7 9
```

●奇数の 2 乗和

```
   m2"0 [5 6 7 8 9              1 で始まる k 個の奇数の和は k の 2 乗になる
25 36 49 64 81
   m6 9                         トートロジー
1
```

●標準偏差の式

```
   m7
%:@(+/@(*:@(] - +/ % #)))       これは平方和の平方根の動詞であり標準偏差ではない
   dev=:%:@((+/@(*:@(] - +/ % #)))%(<:@#@])   標準偏差の動詞
   m7 1 2 3 4 5                 平方和の平方根
3.16228
```

```
   dev 1 2 3 4 5            標準偏差
1.58114
   3.16228%%:4              平方和の平方根を (標本数-1) の平方根で割る
1.58114
```

G 節．　副詞や接続詞をつくるトレイン

`c0=: ([.@{.) , (].@}.)`	始めの x 個のアイテムには f が、残りには g が働く
`c1=: 2 : 'x.@{. , y.@}.'`	c0 の明示的定義
`c2=: 12 : 'x.@{. , y.@}.'`	上と同じだが、黙示的定義に変換可能

●トレインで定義された接続詞と副詞

```
   c0=:([.@{.),(].@}.)      左引数は先頭から、右引数は最後尾からのアイテムにそれぞれ動詞
   c0                       を働かせるための接続詞、黙示的定義
([. @ {.) , (]. @ }.)
   %:c0*:                   先頭からは平方根、最後尾からは 2 乗という動詞を働かせる
%:@{. , *:@}.
   x=:2 3 5 7 11 13 17
   3%:c0*:x                 始めから 3 個のアイテムは平方根、残りには 2 乗が作用する
1.41421 1.73205 2.23607 49 121 169 289
   %:c0*:x                  単項で使うと、先頭だけが平方根、残りは 2 乗が作用する
1.41421 9 25 49 121 169 289
   c1=:2 :'x.@{.,y.@}.'     同じものを明示的に定義
   c2=:12 :'x.@{.,y.@}.'    黙示的定義に変換できるように明示的に定義
   c1
2 : 'x.@{.,y.@}.'
   c2
([. @ {.) , (]. @ }.)       変換された黙示的定義
```

H 節．　動名詞

`m0=: horner=: +'*/`	m0 a,x,b,x,c は (a,b,c)p.x と同じ。ホーナーの算法
`m1=: grid=: +'(*i.)/`	grid b,s,n は b から n まで s ステップのグリッドをつくる
`m2=: case1=: _1:'%:'*:@.*"0`	負、ゼロ、正のときそれぞれ、2 乗、-1、平方根となる
`d3=: sort=: /:~@]'(\:~@])@.[`	左引数が 0、1 で昇順、降順のソート
`v4=: cases=: case1 : sort`	単項で case1、2 項でソートという曖昧項動詞
`a5=: sel=: 1 : '] #~ ] x. {.'`	クイックソートのための副詞
`m6=: qs=: ] ' ($:@(<sel), =sel,$:@(>sel)) @. (1:<#)`	再帰的に定義されたクイックソート
`m7=: (0&, +,&0)^:([ '1:)`	繰り返しを伴う動名詞を使った 2 項係数

```
   x=:3
   horner 1,x,4,x,6,x,4,x,1      (3+1) の 4 乗
```

```
256
   (1,4,6,4,1)p.x                  x=3 のときの (x+1)^4
256
   grid 3 0.5 5                    3 から 0.5 おきに 5 まで
3 3.5 4 4.5 5
   case1 _10 0 10                  負のとき 2 乗、ゼロは-1、正のとき平方根
100 _1 3.16228
   0 sort 4 2 _5 0 5 6 6 _1 6 9    昇順でソート
_5 _1 0 2 4 5 6 6 6 9
   1 sort 4 2 _5 0 5 6 6 _1 6 9    降順でソート
9 6 6 6 5 4 2 0 _1 _5
   cases 4 2 _5 0 5 6 6 _1 6 9     単項では、負のとき 2 乗、ゼロは-1、正のとき平方根
2 1.41421 25 _1 2.23607 2.44949 2.44949 1 2.44949 3
   0 cases 4 2 _5 0 5 6 6 _1 6 9         2 項ではソート
_5 _1 0 2 4 5 6 6 6 9
   qs 4 2 _5 0 5 6 6 _1 6 9        クイックソート
_5 _1 0 2 4 5 6 6 6 9
   ((<sel);(=sel);(>sel))4 2 _5 0 5 6 6 _1 6 9
+---------+-+---------+
|2 _5 0 _1|4|5 6 6 6 9|
+---------+-+---------+
```

先頭の 4 より小さいもの、等しいもの、大きいものに分けてゆく。
これをグループ毎に再帰的に繰り返す。
結局、真中の (=sel) のところへ来て再帰は終了する。

２００２年5月１８日

Ｊのフレーズについて
＿＿＿＿ 第５回 ＿＿＿＿

慶応義塾大学理工学部
竹内寿一郎

第３章　指標付けの規則とアレーの要素の置換え

A 節．　指標付け

指標付けには { が用いられる。
{ のランクは0 _であるからx{y は y 全体に対してスカラーの x が対応する。

●基本的に x のアトムに対して指標付けがなされる。
このとき>x はアトムもしくはリストでなければならない。
j 番目の指標 r のとる範囲は、r=:>j{,>x として i.&.(+n)=:j{$y (&. は n を足して n を引く)
なお、負の指標は実用的には後ろから数えた 1 オリジンの指標であるとして良い。

```
   z=:0{t=:3 3 3$'ABCDEFGHIJKLMNOPQRSTUVWXYZ]'
   z
ABC
DEF
GHI
   (1 0;2 1)
+---+---+
|1 0|2 1|
+---+---+
   (1 0;2 1){z
DH
```

x がリストであるからアトムが 2 個として扱われる。
(1 0) 要素と (2 1) 要素がそれぞれの結果で、その結果としてリストを得る。

```
   y=: i.4 5 6 7
   (<<"(0)0 1 2),(<0 1)
+-------+---+
|+-+-+-+|0 1|
||0|1|2||   |
|+-+-+-+|   |
+-------+---+
   ((<<"(0)0 1 2),(<0 1)){y
56 57 58 59 60 61 62
 0  0  0  0  0  0  0
 0  0  0  0  0  0  0
 0  0  0  0  0  0  0
 0  0  0  0  0  0  0
 0  0  0  0  0  0  0
```

```
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
```

上段はｔの始めの軸から順に０ １ ２を選択したから７次のリスト
下段はｔの０ １を選択したから６ ７のテーブルとなった

```
   (<1 0;_2 0)
+----------+
|+---+----+|
||1 0|_2 0||
|+---+----+|
+----------+
   (<1 0;_2 0){z
ED
BA
```

ｘはアトム。それを開いて、０軸から１ ０ 要素（アイテム）、１軸から_2 ０要素（(1 0) 要素）をとる。

次表は要素を取り出す基本的な方法を示している。

n0=: y=: i.4 5 6 7	例で用いられるアレーの定義
n1=: (<,<3){y	y の第 3 アイテム (形は 5 6 7)
n2=: (<,3){y	y の第 3 アイテム
n3=: (<3){y	y の第 3 アイテム
n4=: 3{y	y の第 3 アイテム
n5=: (<,<_1){y	y の最後のアイテム (y の-1 アイテム)
n6=: (<,_1){y	y の最後のアイテム (形は 5 6 7)
n7=: (<_1){y	y の最後のアイテム
n8=: _1{y	y の最後のアイテム
n9=: (_1 ＋#y){y	y の最後のアイテム
n10=: 0{y	y の最後のアイテム
n11=: (-#y){y	y の最後のアイテム
n12=: 3 0 _2 0{y	y の ３０ _2 ０ アイテム
n13=: i=: ?2 3$0{$y	y の 0 軸に関する 2 3 の指標のテーブル
n14=: j=: ? 1{$y	y の 1 軸に関する指標 (アトム)
n15=: k=: ?7 $2{$y	y の 2 軸に関する指標 (7 次元のリスト)
n16=: (<i;j;k){y	APL の表現では y[i;j;k;]
n17=: (<1;2;3){y	APL の表現では y[1;2;3;]
n18=: (<1,2,3){y	APL の表現では y[1;2;3;]
n19=: (<1 2 3){y	APL の表現では y[1;2;3;]
n20=: (<<i){y	APL の表現では y[i;;;;]
n21=: (<<,i){y	APL の表現では y[,i;;...;]
n22=: (,i){y	APL の表現では y[,i;;...;]

```
   <0 1 2
```

```
+-----+
|0 1 2|
+-----+
   <<"(0)0 1 2
+-------+
|+-+-+-+|
||0|1|2||
|+-+-+-+|
+-------+
```
は同じ指標である。なぜなら指標は r=:>j{,>x で得るからである。
```
   (<0 1 2){y
56 57 58 59 60 61 62
   (<<"(0)0 1 2){y
56 57 58 59 60 61 62
```
次のタイプは 2 重目のボックスが 0 軸を表している。j{,>x において 0 しかないからである。
つまり 0 軸の要素 (アイテム) を 3 個取り出すことになる。
```
      <<0 1 2
+-------+
|+-----+|
||0 1 2||
|+-----+|
+-------+
   $(<<0 1 2){y
3 5 6 7
   $(<<i){y             i はアトムである
5 6 7
   $(<<,i){y            ,i で大きさ 1 のリストにすると大きさは同じでもランクは 1 つ上がる
1 5 6 7
```

●ボックスが 2 重、3 重になった場合
```
   ibbb=:<ibb=:<ib=:<i=:1 _1
   jbbb=:<jbb=:<jb=:<j=:2 1
   ijbbb=:<ijbb=:<ijb=:<ij=:2 2$i,j
   (];i&{;ib&{;ibb&{;ibbb&{)z
+---+---+-+---+---+
|ABC|DEF|F|DEF|ABC|
|DEF|GHI| |GHI|   |
|GHI|   | |   |   |
+---+---+-+---+---+
   ibbb
+--------+
|+------+|
||+----+||
|||1 _1|||
```

```
||+----+||
|+------+|
+--------+
```
3 重ボックスは"それ以外"つまり 1 _1 アイテム以外のアイテムである。
```
   ijb
+----+
|1 _1|
|2  1|
+----+
   ijb{y
|rank error
|   ijb    {y
```
>x はアトムもしくはリストでなければならない。
```
   (];i&{;ijbb&{;ijbbb&{)z
+---+---+---+---+
|ABC|DEF|DEF|ABC|
|DEF|GHI|GHI|   |
|GHI|   |   |   |
|   |   |GHI|   |
|   |   |DEF|   |
+---+---+---+---+
```
ijbbb は 3 重ボックスである。従って 1 _1 2 1 以外のアイテムということになる。

n23=: (<<1 4 2){y	y の 1 4 2 アイテム
n24=: (<<<1 4 2){y	y の 1 4 2 以外の全てのアイテム
n25=: (<<<1 4){y	y の 1 4 以外の全てのアイテム
n26=: (<<<1){y	y の 1 以外の全てのアイテム
n27=: (<<<$0){y	無し以外の全て、つまり、全てのアイテム
n28=: (<<a:){y	全てのアイテム
n29=: (<1 3 2;3){y	APL の表現では y[1 3 2;3;;...;]、但し指標原点はゼロ
n30=: (<(<1 3 2);3){y	y[(i.#y)-.1 3 2;3;;...;] 0 軸から 1 3 2 を除く
n31=: (<(<1 3);3){y	y[(i.#y)-.1 3;3;;...;] 0 軸から 1 3 を除く
n32=: (<(<1);3){y	y[(i.#y)-.1;3;;...;] 0 軸から 1 を除く
n33=: (<(<$0);3){y	y[(i.#y)-.$0;3;;...;] 0 軸からは何も除かない
n34=: (<(<$0);3){y	y[;3;;...;] 0 軸が空、0 軸からは何も除かない
n35=: (<a:;3){y	y[;3;;...;] 0 軸が空、0 軸からは何も除かない
n36=: 4{"_1 y	y[;4;;...;] 1 軸だけ 4 を選択、あとは全て
n37=: (<a:;a:;5){y	y[;;5;...;] 0 軸、1 軸が空、そこからは何も除かない
n38=: 5{"_2 y	y[;;5;...;] 2 軸だけ 5 を選択、あとは全て
n39=: (<1 2){y	0 軸は 1、1 軸は 2、あとは全て
n40=: _2{y	最後尾から 2 番目 (1 オリジン) のアイテム
n41=: (<<<3){y	3 以外のアイテム
n42=: (1 2;3 2;0 _2){y	0 軸から 1 2、1 軸から 3 2、3 軸から 0 _2 の要素を選択

\\

●修正は修正内容、修正する指標、{、修正されるべき対象の順に入力する。

```
   z=:0{t=:3 3 3$'ABCDEFGHIJKLMNOPQRSTUVWXYZ]'
   '*'ib}z
ABC
DE*
GHI
   ib
+----+
|1 _1|
+----+
```

z の 1 _1 要素"F"を"*"で置換える。

```
   ('def',:'ghi')i}z
ABC
def
ghi
```

z の 1 _1 アイテムを'def''ghi' で置換える。

```
   (];i&{;ib&{;ibb&{;ibbb&{)"2 t
+---+---+-+---+---+
|ABC|DEF|F|DEF|ABC|
|DEF|GHI| |GHI|   |
|GHI|   | |   |   |
+---+---+-+---+---+
|JKL|MNO|O|MNO|JKL|
|MNO|PQR| |PQR|   |
|PQR|   | |   |   |
+---+---+-+---+---+
|STU|VWX|X|VWX|STU|
|VWX|YZ]| |YZ]|   |
|YZ]|   | |   |   |
+---+---+-+---+---+
```

t のテーブルに対して操作が行われる。
基本は第 1 行の 5 個のボックスで、これが 3 回繰り返される。

●空について

```
   $(<<'')){z
0 3
   $(<a:){z
0 3
```

結果は空であるが形はテーブルで、第 0 軸の大きさがゼロである。

```
   ]k=:<1 2;a:;0 2 0
+--------------+
|+---+--+-----+|
||1 2|++|0 2 0||
```

```
||   ||||     ||
||   |++|     ||
|+---+--+-----+|
+--------------+
   t;k{t
+---+---+
|ABC|JLJ|
|DEF|MOM|
|GHI|PRP|
|   |   |
|JKL|SUS|
|MNO|VXV|
|PQR|Y]Y|
|   |   |
|STU|   |
|VWX|   |
|YZ]|   |
+---+---+
```

1 軸の選択指標はボックスで囲まれた空である。
しかもこの空は 3 重ボックスであるから"空以外"を意味する。
従って第 1 軸の要素は全て選択される。

●負のランクについて
y のランク指定で (-r) が与えられたとする。
それを正のランクに置き換えると、0>.(#$y)-r つまり、r が y の次元より小さければ次元数− r、
r が次元数より大きければ 0 と同じである。
n36=: 4{"_1 y　　APL での表示では y[;4;;...;]
n38=: 5{"_2 y　　APL での表示では y[;;5;...;]
"1 はランクがリストだから 4{"1 y はy[;;;...;4]、"2 はランクがテーブルだから 4{"1 y はy[;;;...;4;]、

負の場合は次元から引いているので、0 オリジンで先頭から r 番目の軸に相当する。

２００２年6月２２日

Jのフレーズについて
第６回

慶応義塾大学理工学部
竹内寿一郎 (0 1 2 3 4)

第３章　指標付けの規則とアレーの要素の置換え

B 節．　マージ (統合) と修正 (要素の置き換え)

２つの引数 x、y のアイテムを予め決められた適当なブール型リストに従って統合することができる。このとき、ブール型リストの大きさは x ＋&# y でなければならない。また x、y は文字なら文字、数値なら数値のように同じ型で、同じ形 (リスト、テーブルなどサイズは異なっても良い) でなければならない。

●マージ (統合)

```
   x=:>;:'That they hunted from hill'    ;:は Word Formation で Word 毎に Box 化する
   y=:>;:'second time me to plain'
   b=:0 1 1 0 0 1 0 0 1 1
   mrg=:1 : '/:@/:@(x."_){,'             x. に名詞が入って (/:@/:@(x."_)) と{と , でフォーク
   x([;];(,.b)"_;b mrg)y
+------+------+-+------+
|That  |second|0|That  |
|they  |time  |1|second|
|hunted|me    |1|time  |
|from  |to    |0|they  |
|hill  |plain |0|hunted|
|      |      |1|me    |
|      |      |0|from  |
|      |      |0|hill  |
|      |      |1|to    |
|      |      |1|plain |
+------+------+-+------+
   /: 0 1 1 0 0 1 0 0 1 1                ブール型リストを Sort してみる
0 3 4 6 7 1 2 5 8 9
0 についてその場所 (index)、次に 1 についてその場所 (index) のリストを作成
   /: 0 3 4 6 7 1 2 5 8 9                さらにその結果を Sort する
0 5 6 1 2 7 3 4 8 9
小さい index 順になるように、場所の index をつくる
   0 5 6 1 2 7 3 4 8 9{x,y
That
second
time
they
hunted
```

```
me
from
hill
to
plain
```

mrg という副詞に b をつけると動詞になるから、これを参考に動詞 MRG をつくる

```
   b mrg
/:@/:@(0 1 1 0 0 1 0 0 1 1"_) { ,
   MRG=:/:@/:@[{]                    (/:@/:@[) と{と] でフォーク
   b MRG x,y
That
second
time
they
hunted
me
from
hill
to
plain
```

●左引数 b はブール型リストである必要はない。下の例は 0,1,2 からなるリスト

```
   b=:0 2 2 1 0 2 2 2 0 0 1 1 2 1 2 1 1 1 1 1
   y0=:'abcd'[y1=:'123456789'[y2=:'zzzzzzz'
   b MRG y0,y1,y2          0 は y0 から、1 は y1 から、2 は y2 からとってきてマージ
azz1bzzzcd23z4z56789
```

a0=: mrg=: 1 : '/:@/:@(x."_) { ,'	x b mrg y はb に従ってx とy を統合する
m1=: MRG=: /:@/:@[{]	b MRG x,y は上に同じ
d2=: alt=: ,@,.	x とy のアイテムを交互に統合する

● alt を使った例として

```
   alt=:,@,.
   x=:'temr rtes'
   y=:'h axbohr '
   x alt y                交互にとってきてマージする
the marx brothers
```

●右引数 y を index で選ばれた要素に関して、もう一方の引数で置き換える例である。例えば

```
   x=:'ABCD'[y=:'abcdefghij'
   i=:4 2 8 6
   i{y
ecig
   ]z=:x i}y
abBdAfDhCj
```

```
   m=:a.{~(a.i.'A')+i.5 5       ABCD ··· XY までの 25 文字からなる 5 × 5 のテーブル
   ]i=:<"1,.&i.~#m              5,.&i.5 は 0〜4 からなる 2 つ並んだ縦ベクトル
+---+---+---+---+---+           それを横方向にボックス化する
|0 0|1 1|2 2|3 3|4 4|
+---+---+---+---+---+
   x=:'+-*%^'
   m;(i{m);x;x i}m              i{m は m の対角要素を取り出すフレーズ
+-----+-----+-----+-----+
|ABCDE|AGMSY|+-*%^|+BCDE|
|FGHIJ|     |     |F-HIJ|
|KLMNO|     |     |KL*NO|
|PQRST|     |     |PQR%T|
|UVWXY|     |     |UVWX^|
+-----+-----+-----+-----+
```

●右引数に対する指標 (Index) をつくる

```
   IR=:@(i.@$@])              Index of Right Argument 右引数の指標をつくる副詞
   A=:IR}                     右引数の選択された部分を修正するための副詞
   d=:(<0 1)&|:               対角要素を指定する動詞
   '+-*%^'(];d@];]IR;d IR;d IR};d A)m
+-----+-----+--------------+-----------+-----+-----+
|ABCDE|AGMSY| 0  1  2  3  4|0 6 12 18 24|+BCDE|+BCDE|
|FGHIJ|     | 5  6  7  8  9|            |F-HIJ|F-HIJ|
|KLMNO|     |10 11 12 13 14|            |KL*NO|KL*NO|
|PQRST|     |15 16 17 18 19|            |PQR%T|PQR%T|
|UVWXY|     |20 21 22 23 24|            |UVWX^|UVWX^|
+-----+-----+--------------+-----------+-----+-----+
   m  diag m  Indices of m Indices of diag Amendments： 上の結果の説明

   ur=:2 _3&{.     Upper Right Corner 右上段隅を指定する、先頭から 2 行最後尾から 3 列
   (2 3$'+-*%^!')(];ur@];]IR;ur IR;ur IR};ur A)m
+-----+---+--------------+-----+-----+-----+
|ABCDE|CDE| 0  1  2  3  4|2 3 4|AB+-*|AB+-*|
|FGHIJ|HIJ| 5  6  7  8  9|7 8 9|FG%^!|FG%^!|
|KLMNO|   |10 11 12 13 14|     |KLMNO|KLMNO|
|PQRST|   |15 16 17 18 19|     |PQRST|PQRST|
|UVWXY|   |20 21 22 23 24|     |UVWXY|UVWXY|
+-----+---+--------------+-----+-----+-----+
```

●まず、IR で全体の指標を定義する。

副詞 IR の左に指標を選択する動詞 u を置く。取り、落とし、対角要素、などなど。

u IR}もしくは u A によって、動詞により選択された指標の場所を決め、置き換える。

先頭の左引数で置き換わるべき内容を指定する。

[注意]

```
   x d IR m
0 6 12 18 24
```

ここで見られるように IR=:@(i.@$@]) の最後の記号] がよく効いていることが分かる。
つまり、これがないと両側形となり、x $ y の演算が行われてしまうからである。

a3=: IR=: @(i.@$@])	右引数の形に対応した指標を作り出すための副詞
m4=: d=: (<0 1)&|:	テーブルから対角要素を取り出す動詞
m5=: ur=: 2 _3&{.	右上隅 2×3 を選び出す動詞

【問題】　IR の後ろの動詞は常に}であるが、{ の場合どうなるか考えてみよ。
(1)　]IR{m
(2)　x]IR{m
何か答えがかえってくるか、もしくはエラーとなるか、考えなさい。
（正解はこの冊子の 1 ページ目、名前の後ろにある。）

２００２年7月２７日

Jのフレーズについて
第７回

慶応義塾大学理工学部
竹内寿一郎

第４章　検索と選択

A 節．　区間

アトムx が区間 y_1, y_2 の間に入っているかどうか調べるのが 2 項動詞dn である。
x dn "0 1 y で、x が区間y に入っていれば 1、そうでなければ 0 を返す。
区間の数学的表現を以下に示す。

集合表示	区間表示	区間の説明
{x|a<x<b}	(a,b)	開区間
{x|a<x ≦ b}	(a,b]	左半開区間
{x|a ≦ x<b}	[a,b)	右半開区間
{x|a ≦ x ≦ b}	[a,b]	閉区間

集合表示	区間表示	区間の説明
{x|a<x}	(a,_)	a より大きな数
{x|x<b}	(_,b)	b より小さい数
R	(__,_)	有限のすべての実数
{x|x ≦ b}	(_,b)	b 以下の数
{x|a ≧ x}	[a,_)	a 以上の数
{a}	[a,a]	a 1 点のみ
{}	(a,a)	空集合

d0=: OO=: ({.@] < [)*.([< {:@])	x が開区間y 内にあるか？OO
d1=: OC=: ({.@] < [)*.([<: {:@])	x が左半開区間y 内にあるか？OC
d2=: CO=: ({.@] <: [)*.([< {:@])	x が右半開区間y 内にあるか？CO
d3=: CC=: ({.@] <: [)*.([<: {:@])	x が閉区間y 内にあるか？CC
d4=: ＋/"1@d2 (例)2 3 4 5 6 7 8 9 d4 3 8	左半開区間y 内にあるx の数 (出力)5
d5=: 0: e.~ class	x が開区間y 内にあるか？OO
d6=: 0 1"_ e.~ class	x が左半開区間y 内にあるか？OC
d7=: _1 0 "_ e.~ class	x が右半開区間y 内にあるか？CO
d8=: _1 0 1"_ e.~ class	x が閉区間y 内にあるか？CC
d9=: class=: [: ＋/"1 [: * -/ (例) 2 3 4 5 6 7 8 9 class 3 8	Class の結果の -2 から 2 は、左区間外、左開区間内、開区間内、右開区間内、右区間外を示す (出力)_2 _1 0 0 0 0 1 2
d10=: [: ~:/ >:/~	x が左半開区間y 内にあるか？OC
d11=:] >/ .>~ [: ,.~ [	x が左半開区間y 内にあるか？OC

d12=: sgd=: *@(-~/~) (例)2 3 4 5 6 7 8 9 sgd 3 8	差の符号のテーブルを転置。|:*x.-/y. (出力)_1 0 1 1 1 1 1 1 _1 _1 _1 _1 _1 _1 0 1
d13=: 0&=@(＋/)@sgd	x が開区間y 内にあるか？OO
d14=: e.&0 1@(＋/)@sgd	x が左半開区間y 内にあるか？OC
d15=: 0&>:@(%/)@sgd	x が右半開区間y 内にあるか？CO
d16=: >/@sgd	x が閉区間y 内にあるか？CC

●基本的な動詞：開区間、左半開区間、右半開区間、閉区間

```
   x=:i.12[y=:3 8
   x([,OO,OC,CO,:CC)y
0 1 2 3 4 5 6 7 8 9 10 11
0 0 0 0 1 1 1 1 0 0  0  0
0 0 0 0 1 1 1 1 1 1 0  0  0
0 0 0 1 1 1 1 1 1 0 0  0  0
0 0 0 1 1 1 1 1 1 1 0  0  0
   x(CC=OC+.CO)y            トートロジー
1 1 1 1 1 1 1 1 1 1 1 1
   x(d5=OO)y                トートロジー
1 1 1 1 1 1 1 1 1 1 1 1
```

● class を利用した区間　d5〜d8

```
   x,:x class y                        _2.. 左区間外、_1.. 左半開区間内、0.. 開区間内、
 0  1  2  3 4 5 6 7 8 9 10 11            1.. 右半開区間内、2.. 右区間外
_2 _2 _2 _1 0 0 0 0 1 2  2  2
```

●ちょっと読みづらい動詞　d10 と d11

```
   d10=: [: ~:/ >:/~              "以上"のテーブルを作り、"等しくない"を作用させる
   2 3 4 5 6 7 8 9 d10 3 8
0 0 1 1 1 1 1 0
   2 3 4 5 6 7 8 9 >:/~ 3 8       "以上"のテーブル
1 1 0 0 0 0 0 0
1 1 1 1 1 1 1 0
   ~:/2 3 4 5 6 7 8 9 >:/~ 3 8     上と下を比較して、notequal のとき 1、そうでなければ 0
0 0 1 1 1 1 1 0
      d11=: ] >/ .>~ [: ,.~ [      (x,.x) で 2 列のテーブルにして、>/ .>なる外積を使う
   2 3 4 5 6 7 8 9 d11 3 8
0 0 1 1 1 1 1 0
   2 3 4 5 6 7 8 9 ([:,.~[) 3 8     まず、2 列のテーブルをつくる
2 2
3 3
4 4
5 5
6 6
```

```
7 7
8 8
9 9
   (x>3),:(x>8)
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 1 1
   >/(x>3),:(x>8)
0 0 0 0 1 1 1 1 1 0 0 0
```

●差を符号化する sgd を利用した区間

```
   sgd=: *@(-~/~)            入れ替えて差のテーブル (横長)、入れ替えて差をとり符号化
   2 3 4 5 6 7 8 9 sgd 3 8
_1  0  1  1  1  1 1 1
_1 _1 _1 _1 _1 _1 0 1
   d16=: >/@sgd              1行目のアトムが大なるところだけ1
   2 3 4 5 6 7 8 9 d16 3 8
0 1 1 1 1 1 1 0
```

２００２年8月3日～5日 (夏合宿)

Jのフレーズについて
第8回

慶応義塾大学理工学部
竹内寿一郎

第4章　検索と選択

B節.　位置付けと選択 (その１　m0～m20)

まず、いろいろな条件を満たす要素の指標 (IO) または指標群 (ISO) を求める動詞から紹介する。求められた指標はいずれもy の指標である。すなわち、(m y){y もしくは(x d y){y である。

m0=: \: (例)m0 5 4 3 4 5 6	下降順の指標群 (ISO) (答)5 0 4 1 3 2
m1=: {.@\:	y の最大値が初めて現れる指標 (IO)
m2=: i.>./	y の最大値が初めて現れる IO、Hook が働く
m3=: {:@/:	y の最大値が最後に現れる IO
m4=: {.@/:	y の最小値が初めて現れる IO
m5=: {:@\:	y の最小値が最後に現れる IO
m6=: i.<./	y の最小値が初めて現れる IO、Hook が働く
d7=: ~: i. 1: (例)'uyrtyzqperty'd7'uyrtyzrytupo'	x とy の初めて異なる要素の IO (答)6
m8=: # - 1: #.~] = ' '"_	y で後から初めてブランクでない要素の IO
d9=: <./@i. (例)'zqperty'd9'uyrty'	y のアイテムに初めて一致するx の IO (答)4　(r に対するx の中での IO)
d10=: [＋ i.&1@}. (例)6 d10 z=.0 0 0 1 0 1 0 0 1 0 1 1	x アイテム以降初めて 1 が出るy の IO (答)8
m11=:] i. 1:	y の中の初めの 1 の IO
d12=: i.	キィx の中のy のアイテムの IO
d13=: <:@($@]) - |.@] i. [	y の中のx のアイテムの IO
d14=: (<:@(#@[) - |.@[i.])"1 (例)y d14 x[x=.'fsg'[y=.'gsfgtrfsda'	y の中のx の最後の IO (答)6 7 3
d15=: |.@[i.] (例)y d15 x[x=.'fsg'[y=.'gsfgtrfsda'	上に同じだが、後ろからの IO (答)3 2 6
d16=: E. # i.@#@] (例)'NHK'd16'KLNHKUONHK'	x で始まるリストに対するy の ISO (答)2 7
d17=: e.~&, # i.@#@] (例)'KLNHKUONHK'd17 'NHK'	y のアイテムに対するx の ISO (答)0 1 2
d18=: =#[:i.[:#[	x とy の等しいアイテムの ISO
m19=: ＋/{.\:	論理リストy の 1 の ISO。 Hook が働く
m20=: [: m38 m39 (例)m20 2 2$0 1 1 0	高階論理アレーにおける 1 の ISO。 (答) J で次の文の実行結果と同じになる。0 1 ;1 0

● #. の２項動詞について

```
x#.y はyのアイテムの荷重和で、+/w*y となる。ここで、wは Product Scan  */\.}.x,1 である。
   */\.}.(x=:6 5 7 3 4 2),1        6 5 7 3 4 2 の先頭の6は関係なく、何でもよい
840 168 24 8 2 1
   5#.2 3 _6 4       多項式 2x^3+2x^2-6x+4 の x=5 における値。 5 5 5 5 #.2 3 _6 4 に同じ
299
   4 _6 3 2 p.5      多項式の動詞p. による表現
299
   m8=:(# - (1: #.~ (] = ' '"_))        フォークが3個ある
   ]z=.(]=' '"_) y=.'ABC D EF G  '      最初のフォーク
0 0 0 1 0 1 0 0 1 0 1 1
   0 0 0 1 0 1 0 0 1 0 1 1#.1     #. は上で説明した通り
3
   2 3 2 1#.1           1+(1)*1+(1*2)*1+(1*2*3)*1=10
10
   1 0 1 0#.1           1+(0)*1+(0*1)*1+(0*1*0)*1=1
1
   0 0 0 1 0 1 0 0 1 1 1 1#.1     結果は(後ろから連続する1の数) ＋1となる
5
   z #.1                          2番目のフォーク
3
   (#y)-(z#.1)                    最後のフォーク
9
   (m8 y){y             後ろからブランクでない最初の文字はG
G
● xアイテム以降初めて1となる
   d10=:([+(i.&1@}.))        1つのフォーク
   6 d10 z=.0 0 0 1 0 1 0 0 1 0 1 1
8
    (6}.z)i.1           先頭から6個落とし、最初の1の場所の指標は2
2
● Menmber e. と E. の違い
   x=.'f'[y=.'gsfgtrfsda'
   x d13 y                          x が y の中で現れる最後の指標、結果は大きさ1のリスト
6
   d13=:((<:@($@]))-((|.@])i.[])    大きく分けてフォーク。右はフォーク、左は動詞の連続
   x((|.@])i.[)y
3
   x(<:@($@]))y
9
   x=.'fsg'[y=.'gsfgtrfsda'
   d14=:((<:@(#@[])-((|.@[)i.]))"1    d14はxはリストでよく、x,y の引数は逆にする
   y((|.@[)i.])x
3 2 6
   y(<:@(#@[])x
```

```
9
   y d14 x              d14 の中で (<:@(#@[) は (<:@#@[ として括弧は無くともよい
6 7 3
   x=.'gt'[y=.'gsfgtrfsgtda'
   d16=:(E.#(i.@#@])   d16 は E. を用いたフォーク
   x(i.@#@])y
0 1 2 3 4 5 6 7 8 9 10 11
   (x E.y)
0 0 0 1 0 0 0 0 1 0 0 0
   0 0 0 1 0 0 0 0 1 0 0 0#0 1 2 3 4 5 6 7 8 9 10 11
3 8
   x d16 y                      'gt' という並びを順次、指標をずらして探してゆく
3 8
   d17=:((e.~&,)#(i.@#@]))     e. は全てのアトムに対して 1 または 0 を与える
   x=.'gt'[y=.'gsfgtrfsgtda'
   x(e.~&,)y                    y はアレーでもよいが、0 または 1 はリストに対して与えられる
1 0 0 1 1 0 0 0 1 1 0 0
   x d17 y                      e. は x のアトム全て 'g' 't' に対して与えられる
0 3 4 8 9
●x と y の等しいアイテムに対して指標を与える
   d18=:(=#([:i.[:#[))
   x=.'gtdgsgthytag'[y=.'gsfgtrfsgtda'
   x d18 y                      リストの大きさは一致していなければならない
0 3 9
   x=.'g'[y=.'gsfgtrfsgtda'     一方がアトム、他方がリストでもよい
   y d18 x
0 3 8
●高階アレーの指標を与える
   m20=:[:m38 m39
   m39=:($#:((# i.@$)@,))
      a=.?3 3$2
   a
0 1 1
0 0 1
1 0 0
   ,a
0 1 1 0 0 1 1 0 0
   ((# i.@$)@,)a              (,a)#i.$,a   というフックになる
1 2 5 6
   ($a)#:1 2 5 6              テーブル a の指標に直す
0 1
0 2
1 2
2 0
```

```
   m20 a
+---+---+---+---+
|0 1|0 2|1 2|2 0|
+---+---+---+---+
   (m20 a){a
1 1 1 1
   m22=: ([:((</\&.|.)i.1:)(' '"_ ~: ]))"1     始めの右括弧だけフォーク
   ]z=.(' '"_ ~: ])"1 y
1 1 1 0 0
1 1 1 1 1
1 1 0 1 1
1 1 1 1 0
1 1 1 0 1
   |."1 z
0 0 1 1 1
1 1 1 1 1
1 1 0 1 1
0 1 1 1 1
1 0 1 1 1
   </\"1|."1 z
0 0 1 0 0
1 0 0 0 0
1 0 0 0 0
0 1 0 0 0
1 0 0 0 0
   (</\&.|.)"1 z
0 0 1 0 0
0 0 0 0 1
0 0 0 0 1
0 0 0 1 0
0 0 0 0 1
   ((</\&.|.)"1 z)i."1[1
2 4 4 3 4
```

２００２年9月２８日

Jのフレーズについて
第９回

慶応義塾大学理工学部
竹内寿一郎

第４章　検索と選択

B 節．　位置付けと選択 (その２　m21〜m43)

前回の続きで、ここでの動詞をm またはd と書くものとすると、これらの動詞によって得られた IO(指標) もしくは IOS(複数の指標) を用いることにより、y から要素を選択することができる。すなわち、(m y){y もしくは(x d y){y で選択を行う。

結果ががテーブル以上のアレーであれば以下の表では、表示の都合上 (例) のところで,m30 y として答がリストになるようにしている。またy が高階のアレーであればその指標は Box で囲まれたものになるので、これも表示の関係で (例) のところで,>m28 y のようにカンマとオープンを先頭に付与し答がリストになるようにしている。

m21=:] i."1 ’ ’"_ (例)m21 a=.2 3$’a c BC’	テーブルy の行の始めのブランクの ISO (答)1 0
m22=: ([:(</\&.|.i.1:)’ ’"_ ~:])"1 (例)m21 a=.2 3$’ b A C’	テーブルy の行のブランクでない最後の文字の ISO (答)1 2
m23=: <:@(＋/\) i. i.@(＋/) (例)m23 0 1 1 1 0 1	論理リストy の全ての 1 の ISO (答)1 2 3 5
m24=:] # [: i. # (例)m24 0 1 1 1 0 1	論理リストy の全ての 1 の ISO (答)1 2 3 5
d25=: ([: $]) #."1 ([: > [) (例)(<2 1 0)d25 i.3 2 2	Boxed されたx の指標をアレーy をベクトル化 したときの指標に変換する (答)10
d26=: ＋/i. (例)4 d26 5	x で始まるアレーy の ISO (答)4 5 6 7 8
m27=: [: i. # (例)m27 3 2 1 0 _1 _2	リストy のすべての ISO (答)0 1 2 3 4 5
m28=: [: { [: i.&.> $ (例),>m28 i.2 3	アレーy の全ての Boxed ISO (答)0 0 0 1 0 2 1 0 1 1 1 2
m29=: ＋/\@}:@(0:,]) (例)m29 3 1 2 2 5 3	}:(0,y) の累積和 (最後尾の要素は無関係) (答)0 3 4 6 8 13
m30=: $ #: [: i. [: # , (例),m30 i.2 3	アレーy の全ての ISO の指標を縦に並べたもの (答)0 0 0 1 0 2 1 0 1 1 1 2
d31=: {@(;/&i.) (例),>2 d31 3	i.x とi.y の全てのペアからなるテーブルの Boxed ISO (答)0 0 0 1 0 2 1 0 1 1 1 2
d32=: [: m20 [e. [: ,] (例),>’tuqf’ d32’ufgtyres’	リストy に存在するアトムに一致するx の ISO (答)0 1 3
m33=: a."_ i.] (例)m33 ’abcABC’	文字列y 内のそれぞれの文字のa. に対する ISO (答)97 98 99 65 66 67

d34=: i."_1 (例)x=.4 3$'tygkhyt'[y=.4 3$'tty' ,x d34 y	i."<:$$x. に同じ。rix をx のアイテムのランクとする。 一般にx i. y の結果の形状は(-rix)}.$y となる。 i. にランクを付与すると($x)=$y でなければならない (答)0 0 1 3 3 2 0 0 2 3 3 3
d35=: E. i. 1: (例)'AB' d35 'abAB ABA'	リストy でx が始めて生ずる IO (答)2
d36=: <:@(+/@(</))	Index in classes x of y ?????
d37=: i. (例)x i.y[x=.10 3$'tygkhyt'[y=.'tty'	テーブルx に対するリストy の IO (答)2　d34 を参照のこと
m38=: <"1	ISO のテーブルy を Boxed で囲んで指標化する
m39=: $ #: (# i.@$)@, (例) ,m39 2 3$0 1 1 0 1 1	高階アレーy における 1 の ISO(m85 の逆関数) (答)0 1 0 2 1 1 1 2
m40=: m20@m43	数値テーブルを文字化したときの 0 に対する Boxed ISO m43 の結果をリスト単位でボックス化する
m41=: m20@m82 (例)>m41 3 3$5 3 6 8 4 9 6 3 7	テーブルy の鞍点の IO (答)1 1
m42=: <:@(+/\) i. i.@(+/) (例)m42 1 1 1 0 1 0 0 0 1 1	論理リストy の中の 1 の ISO (答)0 1 2 4 8 9
m43=:' 0 '"_ E."1 ' '"_,.],.' '"_ (例),m43 ":2 3$8 0 0 7 0 1 ,>m38 m39 m43 y	数値テーブルを文字化したときの 0 の場所を示す論理数 (答)0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 2 0 4 1 2

●2つのフォークを連続して使う

```
   m22=: ([:(</\&.|.i.1:)' '"_ ~: ])"1
   (' '"_ ~: ])"1 'a c   '
1 0 1 0 0 0
```

(' '"_ ~:])、(</\&.|.i.1:) もフォーク

```
   </\ 0 0 0 1 0 1
0 0 0 1 0 0
```

初めて 1 がでてくる所だけ 1 あとは全て 0

```
(|.0 0 0 1 0 0) i. 1
2
```

m22 では&.|. を使い逆順で操作して戻している

●累積和を利用して増加点の場所を特定する

```
   m23=:<:@(+/\) i. i.@(+/)
   m23 1 0 1 0 0 1
0 2 5
   (<:1 1 2 2 2 3) i. 0 1 2
0 2 5
```

●1 の場所を自分自身で指標選択する

```
   m24=:]#[:i.#
   1 0 1 0 0 1 # 0 1 2 3 4 5
0 2 5
```

●高階アレーの指標 (Boxed) をリスト化されたときの指標に変換する

```
   d25=:([: $ ]) #."1 ([: > [)
   (<2 1)d25 i.5 3
7
```

```
   5 3 #. 2 1                                1+3*2=7
7
   (<2 1 0)d25 i.3 2 2                       3 2 2#.2 1 0    0+2*1+2*2*2=10
10
```

アレーの最高階の次数 (i.5 3 の 5、i.3 2 2 の 3) に関係なく指標は決められる

●y の全てのアトム (要素) の指標一覧、Box で囲まれたもの

```
   m28=: [: { [: i.&.> $
   i.&.> $ i.2 3
+---+-----+
|0 1|0 1 2|
+---+-----+
   {0 1;0 1 2
+---+---+---+
|0 0|0 1|0 2|
+---+---+---+
|1 0|1 1|1 2|
+---+---+---+
   m28 i.2 3
+---+---+---+
|0 0|0 1|0 2|
+---+---+---+
|1 0|1 1|1 2|
+---+---+---+
```

●y の全てのアトム (要素) の指標を縦に並べたもの (形状はテーブル)

```
   m30=: $ #: [: i. [: # ,
   2 3#:0 1 2 3 4 5
0 0
0 1
0 2
1 0
1 1
1 2
```

●i.x と i.y の全てのペアからなる指標をつくる

```
   d31=: {@(;/&i.)
   2(;/&i.)3
+---+-----+
|0 1|0 1 2|
+---+-----+
   2 d31 3
+---+---+---+
|0 0|0 1|0 2|
+---+---+---+
|1 0|1 1|1 2|
+---+---+---+
```

```
  3(;/&i.)2 2
+-----+---+
|0 1 2|0 1|
|     |2 3|
+-----+---+
   3 d31 2 2
+---+---+
|0 0|0 1|
+---+---+
|0 2|0 3|
+---+---+

+---+---+
|1 0|1 1|
+---+---+
|1 2|1 3|
+---+---+

+---+---+
|2 0|2 1|
+---+---+
|2 2|2 3|
+---+---+
```

● y の中にある x を選択するための指標をつくる

```
   d32=: [: m20 [ e. [: , ]
   'tuqf'([: m20 [ e. [: , ])'ufgtyres'
+-+-+-+
|0|1|3|
+-+-+-+
   ('tuqf' d32 'ufgtyres'){'ufgtyres'
uft
```

● i. はランクに注意

ランク_1 は (最大ランクー１) に同じ。すなわちアイテムを対象とする

なお、i. にランクを付与すると x. と y. の形状は同じでなければならない

```
   d34=: i."_1
   x=.3 3$'tyjhgjhyt'[y=.3 3$'jythgjfhu'
   x i. y
3 1 3                          テーブル x に対して結果として各リストの指標を得る
```

i. のランクは (_ _) であるが、x のアイテムに対する y の指標を見つける

```
   x=.3 3 3$1 2 3
   x i. 3 3$1 2 3
0
   x i. 3 3$1 1 1
3                              答の 3 は範囲外であることを示す
```

```
   x i."_1 y                    xのアイテム(リスト)に対するyのアトムの指標を得る
2 1 0
0 1 2
3 0 3
```
● ********* d36の使用法、不明 **********
```
   d36=: <:@(+/@(</))
   1 4 2 3 d36 1 1 2 3 3 4 6 6 6
_1 _1 0 1 1 2 3 3 3
   1 1 2 3 3 4 6 6 6 d36 1 4 2 3
_1 4 1 2
```
● i. は初めて出現する指標を返す
```
   m42=: <:@(+/\) i. i.@(+/)          (左が1の累積和)、i.、(右が1の個数)、のフォーク
   <:(+/\y)                           1があるところで1ずつ増加する
0 1 2 2 3 3 3 3 4 5
0 1 2 2 3 3 3 3 4 5 i. 0 1 2 3 4 5    1の個数それぞれの位置を決める
0 1 2 4 8 9
```
● m43はm39のためのテーブルをつくる
```
   m43=:(' 0 '"_ E."1 (' '"_ ,. (] ,. ' '"_)))      3個の連続したフォークからなる
   y=.":2 3$8 0 0 7 0 1
      (' '"_ ,. (] ,. ' '"_))y        先頭、最後尾の列にブランク列を付加する
 8 0 0                                ' 0 '"_ E."1は行方向に' 0 'を順次探索して行く
 7 0 1
   m43 y
0 0 1 0 1 0 0
0 0 1 0 0 0 0
   m39
$ #: (# i.@$)@,
   m38
<"1
   m38 m39 m43 y
+---+---+---+
|0 2|0 4|1 2|
+---+---+---+
```
● m39の復習
```
   ]d=:m43 y
0 0 1 0 1 0 0
0 0 1 0 0 0 0
   ,d
0 0 1 0 1 0 0 0 0 1 0 0 0 0
   i.$0 0 1 0 1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13
   (,d)#0 1 2 3 4 5 6 7 8 9 10 11 12 13  (# i.@$)はフック
2 4 9
   ($d)#:2 4 9                        7進法により表現する。全体としてフォーク
```

```
0 2
0 4
1 2
```

２００２年１０月２６日

Ｊのフレーズについて
第１０回

慶応義塾大学理工学部
竹内寿一郎

第４章　検索と選択

Ｂ節．　位置付けと選択 (その３　m44〜m59)

以下は指標ではなく選択すべき要素を抽出する論理アレーをつくる動詞群である。
表中の動詞をm、ｄとすると、選択は(m y)#y または(x d y)#y で実行することにより、目的とする要素を選択することができる。

m44=: =&’’’’ (例) m44 ’n’’’’qp’’njif’’j;nj’	y の中の引用符の場所 (答) 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0
m45=: = >./	y の中の最大値の場所全て
m46=: ’.’&~:	y の中のピリオドを除く
m47=: e.&’ 0123456789’	y の中のスペースまたは数字の場所
d48=: E. (例)’nj’ d48 ’nurqp’’nji...j;nj’	パターンx で始まるy の場所 (答)0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
m49=: 2&=@(＋/@(0&=@(|/~@i.))) (例)(m49 20)#i.20	y より小なる素数の場所 (答)2 3 5 7 11 13 17 19
d50=: i.@(#@])e.[(例) (5 2 3 7 d50 7 3 4 2 1 5)#i.6	i.#y でつくられるアトムがx の中にあるかどうか (答)2 3 5
d51=:] e.~ [: i. [: # [	指標i.#x がy のアトムの中にあるかどうか
m52=:] e.~ [: i. [: >: >./	0 からy の最大値までの正の整数がy の中に存在するかどうか
m53=: *./ .=&’ ’ (例)m53 5 3$’jiu　jh　kijacd’	テーブルy のブランク行の場所 (答)0 1 0 1 0
m54=:] -:"1 [: {. ’ ’"_ ,] (例)m54 3 3$’ab　21’	テーブルy のブランク行の場所 (答)0 1 0
m55=: ~: (例)~:’764567675486’	最初のアイテムが出現する場所 (答)1 1 1 1 0 0 0 0 0 0 1 0
m56=: [: *./\ ’ ’"_ =] (例)m56 ’　jh　kijacd’	先頭から連続するブランクの場所 (答)1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
m57=: [: *./\. ’ ’"_ =]	最後尾から連続するブランクの場所
m58=: 2:＋./\0:,2:|＋/\@(=&’’’’) (例)m58 ’a jh’’ki jacd’’ gh’	引用符で囲まれたテキストの場所 (引用符も含む) (答)0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0
m59=: 2:*./\0:,2:|＋/\@(=&’’’’) (例)m59 ’a jh’’ki jacd’’ gh’	引用符で囲まれたテキストの場所 (引用符は含まない) (答)0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0

●素数の取り出し m49

```
   m49
2&=@(+/@(0&=@(|/~@i.)))
```

```
   |/~@i.6
0 1 2 3 4 5
0 0 0 0 0 0
0 1 0 1 0 1
0 1 2 0 1 2
0 1 2 3 0 1
0 1 2 3 4 0
```
i.y の剰余表を作成して 1 とそれ自身の剰余がゼロのとき素数とする。
従って縦に見て 0 が 2 個あれば素数であると判定する

●リストからアトムを選択 d50、d51

```
   d50
i.@(#@]) e. [
   5 2 3 7 d50 'abcdef'
0 0 1 1 0 1
   0 1 2 3 4 5 e. 5 2 3 7
0 0 1 1 0 1
   (5 2 3 7 d50 'abcdef')#0 1 2 3 4 5
2 3 5
```
d50 は 0 から#y 未満内にある y のアトムを選択する指標をつくる

```
   d51
] e.~ [: i. [: # [
  'uytrt'd51 4 6 5 3 1
0 1 0 1 1
   0 1 2 3 4 e. 4 6 5 3 1
0 1 0 1 1
 ('uytrt'd51 4 6 5 3 1)#4 6 5 3 1
6 3 1
```
d51 は 0 から#x 未満内の指標が y のアトムの中にあるかどうか調べる。その論理リストから得られる結果はまさに y の並び順に強く依存することに注意する必要がある。
なお、d50 では y. の、d51 では x. の大きさ (サイズ) だけが関係する

●テーブルのブランク行を選択する m54

```
   m54
] -:"1 [: {. ' '"_ , ]
   y=.3 3$'ab     21'
   ${.(' '"_ , ])y          ブランク行の大きさを調べる
3
   y-:"1{.(' '"_ , ])y
0 1 0
```
ブランク行のサイズが分からないので先頭にブランクを付加してそれを使用する。
テーブルの先頭にブランクを付加すると、行の大きさを持つブランク行が作成される

●前からまたは後ろからのブランクを選択 m56、m57

```
   m56
[: *./\ ’ ’"_ = ]
   y=.’   jh    kija  ’
   (’ ’"_ = ])y
1 1 1 0 0 1 1 1 1 0 0 0 0 1 1
   *./\1 1 1 0 0 1 1 1 1 0 0 0 0 1 1
1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
   *./\.1 1 1 0 0 1 1 1 1 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
```

./\は前からの論理積、./\. は後ろからの論理積

●’で括られたリストの選択 m58、m59

```
   m58
2: +./\ 0: , 2: | +/\@(=&’’’’)
   m59
2: *./\ 0: , 2: | +/\@(=&’’’’)
   +/\(=&’’’’)’ab’’cde’’fg’’ijkl’’mn’
0 0 1 1 1 1 2 2 2 3 3 3 3 3 4 4 4
```

奇数が概ね’で囲まれたリストである。’で囲まれれば必ず奇数は２個以上続く。
このリストについて２の剰余をとり論理リストに変え、先頭に０を付加する。
その後2+./\または2*./\の演算を行う。
m v/\ n はｎのアイテムをｍ個ずつ、１つずつずらしながらｖの挿入演算を行う。
ちなみにｍがマイナスのときは１つずつずらさず、重複がないようにずらす。

この最後の演算について詳しく検討してみよう

```
   2+./\1 0 1 0 1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1
   2+./\0 1 0 1 0 1 0 1 0 1 0 1 0
1 1 1 1 1 1 1 1 1 1 1 1 1
   2*./\1 0 1 0 1 0 1 0 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0
   2*./\0 1 0 1 0 1 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0
```

論理値０ １が交互にくると+. では全て１、*. では全て０になる。
まず先頭の２個のアトムから演算を開始するので全体の大きさが１減ずることに注意。
このために、0: , として先頭に０を付け足し、大きさを揃えている

```
   2+./\1 1 0 0 1 1 0 0 1 1 0 0 1 1
1 1 0 1 1 1 0 1 1 1 0 1 1
   2+./\0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 1 0 1 1 1 0 1 1 1 0
```

+./\では０から１、１から０に変わるところで１になる

```
   2*./\1 1 0 0 1 1 0 0 1 1 0 0 1 1
1 0 0 0 1 0 0 0 1 0 0 0 1
   2*./\0 0 1 1 0 0 1 1 0 0 1 1 0 0
```

```
0 0 1 0 0 0 1 0 0 0 1 0 0
```
*./\では 1 から 0、1 から 0 に変わるところで 0 になる

```
   2+./\1 1 1 1 0 0 0 0 1 1 1 1
1 1 1 1 0 0 0 1 1 1 1
   2+./\0 0 0 0 1 1 1 1 0 0 0 0
0 0 0 1 1 1 1 1 0 0 0
   2*./\1 1 1 1 0 0 0 0 1 1 1 1
1 1 1 0 0 0 0 0 1 1 1
   2*./\0 0 0 0 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 0 0 0 0
```
m58 は’も含めて選択、m59 は’をカットして選択する

実行例：

```
   (>@(m58;m59) y)#"1 y=:’ab’’cde’’fg’’ijkl’’mn’
’cde’’ijkl’
cdeijkl
```