APLとOR(1)

竹下　亨

1. はじめに

計算機についてまったくの素人の人は，APLを数学の記法の延長として，少しずつ計算機と対話しながら，その使い方を覚えるのが手取り早いとされている．FORTRANやCOBOLやPASCALなどを知っている人たちには，APLはなじみにくい言語かもしれないが，慣れれば慣れるほど，適用の幅が広く，また奥が深いことがわかってくる．

APLの特徴は，演算の対象を配列（スカラーと多次元を含む）としており，それに関する便利な関数や作用子を備え，演算の手順が簡潔に表現できることである．APLの文は密度が高く，他言語で数個の文に相当することが1つの短い文ですむことが少なくない．

8，9月号では「APL」言語を紹介することになっているが，8月号ではAPLの生い立ちと特徴から始めて，この言語で扱えるデータの種類，組み込まれている関数や作用子，配列の要素の選択，関数の定義の仕方，入出力，実行順の制御など基本的な書き方や読み方を眺めてみよう．

そして，次号では，ORの問題を解くのに必要なプログラミングの観点から，具体的に簡単な数値計算や作表やグラフの作成を試みよう．初歩的なプログラムの作り方がわかったところで，APLの適用分野や標準化の動き，今後の見通しにふれよう．終りにAPLの言語やプログラム作成をくわしく勉強されたい読者のために，参考文献をあげておいて，実際の応用例の執筆者にバトン・タッチしたい．

2. APLの生い立ち

APLの原形は，1956年にハーバード大学にて自動データ処理の大学院コースを担当していたK. E. Iversonによって，データ処理のさまざまな問題の記述と分析に，また教育および教材作成に考案された記法であり，当初はIverson記法とよばれた．間もなく他の人たちもその言語の適用と開発に興味をもち，多種の問題に利用されて，その価値が認められることになった．

1960年にIversonはIBMのT. J. Watson研究所に移ってから，A. D. Falkoffが自分のプロジェクトにAPLが役立つことを見出し，1962年にはIversonとE. H. Sussenguthとともにシステム/360の形式的記述の仕事を始めた．

APLを計算機で翻訳・実行させる試みは1964年に始められ，翌年にはIBM 7090でバッチの処理系が作られた．そして，1966年にシステム/360で時分割方式の実験的システムが開発され，それが製品として1968年に客先に提供された．それにカードやテープなどの装置による入出力や他のユーザーとデータを共用する機能をもつ処理系（APLSV）が1973年に，また現在広く使われている系統の処理系（VS APL）の第1版が2年後に姿を現わした．現在では国産（富士通，日立，日電，三菱など）を含め主要メーカーの多くの機種で使えるようになっており，ミニコンやマイコンでも使える機種がある．ユーザーの数は日本では数百社ある．

3. APLの特徴

APLは，数値的および論理的関係を簡潔かつ厳密に表現するように考え出されたものであるが，大きくまとめて次のような特徴をもっている．

(1) 言語の演算の対象は配列であり，$A+B$は配列AとBの対応する要素間の加算を意味する．Aの形（各次元の長さ）はρAで示され，配列の要素は指標（添字）をつけることにより表わされる．

(2) 構文は単純であり，文の種類は，代入，分岐，その他の3つである．関数に優先順序がなく，常に右から左に演算される．関数には，0から2個までの引数（ひきすう）をともなう．

(3) 意味上（semantic）の規則も少ししかない．原始

たけした　とおる　日本アイ・ビー・エム㈱

関数は，引数のデータの表現に独立に定義されている．スカラー関数は配列の対応する項目（要素）間に拡張される．原始関数には副作用がない．

(4) 実行順の制御が簡単である．1種類の文がすべての型の分岐(無条件，条件，計算型など)をカバーできる．ユーザー定義関数が実行されるとそれを呼び出したところに制御がもどる．

(5) ＡＰＬプログラムの外部との連絡は，ＡＰＬと他のシステムやサブシステムとの間で共用される変数によって行なわれる．共用変数の部分集合であるシステム変数は，ＡＰＬプログラムとシステム環境との連絡手段となっている．

(6) 体系的に関数を修飾する作用子があり，原始関数の実用性が大きく拡張される．たとえば，簡約によって，関数を並びの全要素に適用せしめる．

ＯＲの計算で，表や行列の形のデータを扱うときにはこれらの特徴からＡＰＬが好都合なことが少なくない．以下これらをやや具体的に説明していくことにしよう．

4. データの種類

ＡＰＬでは，数値定数と文字定数が扱われる．

定数 { 数値定数 { 標準形式：通常の10進表現 / 指数形式：通常の10進表現に E_n をともなうもの } / 文字定数 }

ＡＰＬでは，負の符号に¯（上線記号）を使い，引き算を表わす－（横棒記号）と区別している．

〔例〕

```
128  ¯5.3  23E5  3.1E¯8
```

文字定数は，1字だけのときは文字スカラー，空または2字以上のときは文字ベクトルを意味する．これらは引用符で囲まれる．文字ベクトルの中の引用符は2つ並べて表わす．

〔例〕

```
'A'  'CAN''T'  ''
```

名前は英字で始まり，空字を含まない英数字の組合せで作られる．ある変数に値を代入するには，

変数←定数

の形を使う．これを代入文という．（なお，文は左端から7字目から始まる．）

〔例〕

```
DLT←0.005……正の数値スカラー
V←3  8  5……数値ベクトル
TEMP←¯3.8……負の数値スカラー
NM1←'TARO'……文字ベクトル
W←''……空の文字ベクトル
```

表 1　スカラー原始関数

記号	1項形式	2項形式	記号	1項形式	2項形式
＋	共役	加算	○	円周率倍	三角関数
－	逆符号	減算			双曲線関数
×	符号関数	乗算			ピタゴラス関数
÷	逆数	除算	！	階乗	2項係数
｜	絶対値	剰余	～	論理否定	
⌊	床	最小	∧		論理積
⌉	天井	最大	∨		論理和
？	乱数	抜取乱数	⍲		否定論理積
＊	指数関数	累乗	⍱		否定論理和
⍟	自然対数	一般対数			

記号	2項形式	
＜	小さい	関係（比較）関数
≤	小さいか等しい	
＝	等しい	
≥	大きいか等しい	
＞	大きい	
≠	等しくない	

5. 基本関数

APL に組み込まれている原始関数 (primitive function) を表わす44個の記号があり，ほとんどが1項と2項の関数に使われている．たとえば，－の記号は逆符号と減算を表わす．

通常の算術演算の外に，初等関数や行列演算などの基本関数すなわち原始関数が数多くあるが，これらについては，表1を参照されたい．数学の記法にほぼしたがうが，絶対値は縦棒1本で表わされ，右側に引数がおかれる．

数学と異なる点は，前述のように演算の優先順位がなく，複数個の関数が含まれていると，右から左へ演算が行なわれる．このことに初めは抵抗を感じる人たちが少なくないが，このやり方により，かっこは少なくてすみ，原始関数と定義関数を一様に取り扱うことができる．また数多い関数の優先順位を覚えなくてもよい．

スカラー原始関数の中で，関係関数（比較関数ともいう）は，真であれば1が，偽であれば0が求まる．論理関数の引数とその結果は1(真)か0(偽)である．

〔例〕

```
      3  5＜2  8……演算の結果は次の
0  1……ベクトルは左端から表示 行に表示される．
```

ＡＰＬのプログラムで使われる配列の形を見るには，ρ(ロー記号)で表わされる形(shape)関数を使う．

表 2　原始混合関数

関数名	記号	関数名	記号
形	ρA	指標生成子	ιS
変形	$V\rho A$	指標調べ	$V\iota A$
ベクトル化	$,A$	所属関係	$A\varepsilon A$
逆順	ΦA	昇順	ΔV
回転	$A\Phi A$	降順	∇V
連結	A, A	抜取乱数	$S?S$
転置	$V\circledcirc A$	逆行列	$\boxplus A$
	$\circledcirc A$	行列除算	$M\boxplus M$
取り	$V\uparrow A$	復号	$A\perp A$
落し	$V\downarrow A$	符号	$A\top A$
圧縮	V/A	実行	$\circledast V$
膨張(拡張)	$V\backslash A$	1項書式	$\circledcirc A$
指標付け	$V[A]$	2項書式	$\circledcirc VA$
	$M[A;A]$		
	$A[A;..;A]$		

〔例〕

```
          ρ5  3  2……{引数は3つの要素より
3……                 {なるベクトル
```

ベクトルから配列を作るには，同じ記号で表わされる2項関数の変形(reshape)関数を使う．

〔例〕

```
          M←2  3ρ5  6  7  8  3  2
5  6  7 ……{12個の要素からなるベクトルから
8  3  2    {2×3の行列が作られた．
```

M に2を掛けると，次のようになる．

```
          M×2……M の各要素に2が掛けられる．
10  12  14
16   6   4
```

以上の例でわかるように，新しい文は左端から7字目から始まるようにシステムが自動的にスペースする．またベクトルや文字配列は左端から表示される．数値の行列やそれ以上の次元の配列は，左端に1字分か2字分のスペースをおく．

6. 作用子

APLでは，関数を修飾して，新しい関数を派生させる作用子(operator)があり，簡潔に表現する能力をさらに向上させており，APLの重要な特色となっている．たとえば，／(斜線)で表わす簡約(reduction)を加算＋に作用させると，配列の各行の合計が求まる．

〔例〕

```
          +/3  5  6  7
21……3＋5＋6＋7の結果である．
```

簡約／を乗算×に作用させると，要素を全部掛けたものを，最大「に作用させるとベクトルの中の要素の最大値が求まる．

〔例〕

```
          ×/3  1  4  2
24……3×1×4×2の結果である．
          ⌈/3  1  4  2
4  ……3⌈1⌈4⌈2の結果である．
          P←5  7  6  8……単価
          Q←3  0  4  2……個数
          P×Q              ……品目別金額
15  0  24  16……対応する要素を掛けた結果
          +/P×Q           ……合計額
55
```

簡約はある集合を代表するような数値すなわち，合計，最大値，最小値，平均値などを求めるのに便利である．簡約と類似しているが，左端からその位置の要素までの簡約の演算結果を新しい要素とする走査 (scan) があり＼(逆斜線)で表わす．

〔例〕

```
          +\3  2  1  5
3  5  6  11……たとえば3番目は3＋2＋1
```

走査は，その日や月や年までの合計，最大，最小などを求めるのに利用される．

2つの記号 ∘ . (小丸と点)で表わされる外積 (outer product) という作用子があり，左側の引数の各要素が右側の引数の各要素と対になって，その間に原始関数の演算が行なわれる．

〔例〕

```
          P←1  2  3
          Q←1  2  3  4
          P∘.×Q
1   2   3   4    5……1×1  2  3  4
2   4   6   8   10……2×1  2  3  4
3   6   9  12   15……3×1  2  3  4
          P∘.<Q
0   1   1   1   1……1<1  2  3  4
0   0   1   1   1……2<1  2  3  4
0   0   0   1   1……3<1  2  3  4
```

また，行列の内積を拡張して，対応する行の要素と列の要素の積(×)の和(＋)の代りに，他の2項原始関数もおけるようにした内積(inner product)という作用子があり，2つの原始関数を f と g とすると，$f.g$ で表わされる．

〔例〕

Pは 2 3 5 7，Mは

$$\begin{matrix}1&1&1&1\\0&1&1&1\\0&0&1&1\\1&0&0&1\end{matrix}$$

とすると，その内積は次のようになる．

$M+.\times P$

17 15 12 7……たとえば15=(0×2)+(1×3)+(1×5)+(1×7)

このほかの作用子に軸(axis)があり，大かっこで囲まれた式で表現される．これは2次元以上の配列について簡約，走査の作用子が働く方向や配列の逆順，連結，圧縮，拡張，回転の関数の働く方向を示す．なお，軸作用子の指定がない場合は，最後の軸に沿って行なわれる．

〔例〕

1から12までの自然数により構成される3×4の行列Mがあるとする．

$+/[1]M$

15 18 21 24

$+/[2]M$

10 26 42

$+\backslash[1]M$

1 2 3 4
6 8 10 12
15 18 21 24

$+\backslash[2]M$

1 3 6 10
5 11 18 26
9 19 30 42

7. 要素の選択

空やスカラーではない配列の中の要素を選ぶには，各次元(軸)に対して指標(index)をつける．指標は配列名の後に大かっこで囲んである．複数の軸があるときは，セミコロンで区切る．指標そのものがベクトルや行列などのことがある．

〔例〕

$V\leftarrow 3\ 8\ 5\ 6$

$V[3\ 1]$……Vの3番目と1番目の要素

5 3

$A\leftarrow$'ABCDE'

$A[3\ 1\ 4]$

CAD

Mを1から8までの自然数を要素とする2×4の行列とする．

$M[2;3]$

8

$M[2\ 1;2\ 3\ 4]$

6 7 8
2 3 4

なお，$M[I;]$と$M[;J]$はそれぞれすべての列や行が選ばれることを示し，

$M\leftarrow[2;]$……2行目のすべて(の列)

5 6 7 8

$M\leftarrow[;2]$……2列目のすべて(の行)

2 6

APLには，上記の指標付け (indexing) のほかに，配列の特定要素を取り出す関数に，取り (take) と落し (drop) の2つ，圧縮 (compress) と拡張 (expand) の2つがある．

Sが負でないスカラー整数とし，Vがベクトルであれば，$S\rightarrow V$はVの頭からS個の要素を取り出す．Sが負数であれば，Vの終りから取り出す．

〔例〕

$V\leftarrow 1\ 2\ 5\ 8$

$3\uparrow V$……頭から3つ取り出す

1 2 5

$-2\uparrow V$……終りから2つ取り出す

5 8

逆に，頭から終りから何個かの要素を落としたいときは，$S\downarrow V$を使う．

〔例〕

$2\downarrow V$……頭から2つ落とす

5 8

$-1\downarrow V$……終りから1つ落とす

1 2 5

圧縮の関数は，／で表わされ，残す文字位置と除く文字位置をそれぞれ1と0で表わす論理ベクトルUにしたがって，ベクトルXを圧縮する．

〔例〕

$X\leftarrow 2\ -5\ 3\ -8$

$U\leftarrow 1\ 0\ 1\ 0$

U/X

2 3

圧縮とは逆に，数値ベクトルに0の要素を挿入したり，文字ベクトルに空字を挿入するのに，＼で表わされる膨張という関数を使う．

〔例〕

$V\leftarrow 2\ 3\ 8$

$U\leftarrow 1\ 0\ 0\ 1\ 1$

$U\backslash V$

2 0 0 3 8

8. 関数の定義

APLでは端末でサイン・オン(ログ・オン)したときは，実行モードにあり，APL文を1つ入れるごとに，解釈実行される．ユーザーが一組の文を関数として定義しておいて，くり返し，またはいくつかのプログラムで

共通に使うことができる．

関数を定義するのに，▽(逆三角記号)をキー・インして，関数の見出しを入れる．それは引数の個数（0か1か2）や結果変数（単数）の有無を明示する．その形は次のとおりである．左側は結果変数がない場合である．

F	*R*←*F*
F B	*R*←*F B*
A F B	*R*←*A F B*

関数の見出しを入れると，次の行の左端に[1]が表示されて，ユーザーが次の文を入れるのを促し，それが入ると，[2]が表示されて，2番目の文が入るのを待つ．というように続き，関数の終りは逆三角形を入れて，定義モードから実行モードにもどる．

〔例〕

平均を求める関数MEANは次のように定義される．

```
      ▽MEAN  X
[1]   (+/X)÷⍴X
[2]   ▽
```

これを使って，1から5までの整数の平均を求める．

```
      MEAN  1  2  3  4  5
3
```

ユーザーが定義する関数の中で，すでに定義された関数を使うことができる．また再帰的関数を定義することも可能である．

〔例〕

階乗を求める関数は次のように定義できる．

```
      ▽Z←FAC  X
[1]   →(X=0)/END
[2]   Z←X×FAC  X-1 ……{自分自身を呼び出す
[3]   END:Z←1▽
```

定義された関数の全体や一部分を表示したり，特定の文の追加，挿入，削除，また文の中の文字の追加，挿入，置換えなどを行なうことができる．

9. 入出力

変数に値を与えるには，その名前と左矢印とその右側に与えるべき定数か計算すべき式をおくことは，すでにみた．APLでは，変数名だけをおくと，数値でも文字でもそのときの値を表示する．また，左矢印の左側に□(四角記号)をおくときには，右側に与えられた値が表示される．

〔例〕

```
      X←5  6
      X……Xの値を表示する．下の特別の場合．
5  6
      □←X……Xの値を表示する．
5  6
```

文字のテキストを出力するには，それを'(引用符)で囲んだものを1つの文として与えてもよい．

```
      'SAMPLE  TEXT'
SAMPLE  TEXT
```

関数の実行中に数値を入力するには，式の中に□をおく．これにより評価された入力が行なわれ，定数でも式でもその場で入れることができる．

〔例〕

```
      RADIUS←□
      AREA←(6×□)÷2
```

左矢印の右側に四角記号があると関数の実行を停止して，次の行の左端に，

```
□:
```

を表示し，1行送られて，6字分右にスペースして，ユーザーからの入力を待つ．

文字のデータを入力させるには，⍞（四角引用符号）を左矢印の右側におく．これに遭遇すると，実行を中断し，1行送って，左端からの入力を待つことになる．

APLでは，標準の出力書式で表現する場合は，書式を与える必要がない．しかし，数値データを文字データに変換したり，数値配列の出力の形態を指定するのに，それぞれ1項と2項の書式 (format) 関数を使い，いずれも⍕(昼根小丸記号)を使う．後者は，書式を*F*，配列を*A*とすると，*F*⍕*A*で表わされる．*F*は単一の数であったり，1対の数であったり，2×¯1↑1，⍴*A*を長さとするベクトルでもよい．

一般に，1対の数の場合は，書式は結果の制御用に使用される．1番目の数で数字フィールド全体の幅を定め，2番目の数で精度を指定する．10進数書式の場合に精度は小数点の右側の桁数として指定される．指数形式の場合に仮数の桁数として指定される．指数形式のときは，精度を表わすほうに負の符号がつけられる．

〔例〕

```
      A←12.34
      9  2⍕A
 12.34
      9  ¯2⍕A
 1.2E01
```

以上で，APLの特徴の概略と言語にそなわっている関数と作用子，ユーザーが関数を定義する方法，入出力の指定のやり方を紹介した．次号ではORの計算にも出てくる初歩的な具体例を少々試みることにしよう．