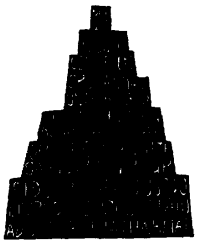

11. A　P　L†

竹　下　　亨††

1. はじめに

APL (A Programming Language の略) は計算機で処理させたいさまざまな問題の数値的および論理的関係を厳密かつ簡潔に記述するための表記法 (notation) として，1956年頃から K. E. Iverson により考え出された．彼は当時 Harvard 大学の計算研究所の Howard Aiken 教授の下で助教授の職にあり，1960年に IBM T. J. Watson 研究所に移った．その著書 "A Programming Language" の序文の中で，「種種の関数の解を得るための明示的な手順などは算法 (algorithm) もしくはプログラムと呼ばれる．それはかなりの構文的構造を示すので，プログラミング言語と称する」と述べられている．

当初は計算機で処理する数値解析，分類などの算法の記述やさらに計算機の内部論理 (logic) の記述や APL 言語そのものの形式的記述に使われた．また大学や高等学校で教育にも利用され，前述の書物や文献[4] がまとめられた．やがてこの言語で書かれたプログラムを対話式に解釈実行する処理系が開発され，実用化されたが，その発展経過の概略は**表-1**の通りである．

今日では，Burroughs, CDC, DEC, Hewlett Packard, Honeywell, IBM, Univac などアメリカのメーカおよび富士通，日立，日電，三菱の国産各社により提供されている．さらに，ミニコンやマイコンにも APL の処理系が作られている．また北アメリカでは I. P. Sharp 社や Scientific Time Sharing 社など約30社により APL の TSS サービスが提供されている．

2. 最近の動向

APL の設計思想については，文献[7),9)]に記述されている，この言語の開発上の主たる思想や指針とする大原則は，大別すると単純性 (simplicity) と実用性 (practicality) であると設計者は述べている．

（1） 単　純　性

（a） 一様性 (uniformity)：規則はごくわずかで，しかも一般的に適用される．たとえば，加減乗除などスカラの引数に定義されたものが，そのまま2次元以上の配列に使われる．

（b） 一般性 (generality)：少数の関数や作用子がその特別の場合として，特殊な関数や特殊な作用子の働きをする．たとえば，簡約は，並んだ項目の累計のほかに，全項目の積や最大，最小などの関数の働きを派生させ，外積は×以外に，内積も×と＋以外の関数に拡張されている．

（c） 親近さ (familiarity)：基本的には，一般的な数学的記号の使い方をベースとして，なるべく親しまれている記号や使い方を採用している．

（d） 簡潔さ (brevity)：表現の経済性を追求して，少ない文字で表現する．

（2） 実　用　性

表-1　APL の発展経過

年	事　　項
1956	K. E. Iverson が APL の記法を考案
1962	A Programming Language を出版
1963	IBM 1620 で試験的に実働化
1965	Stanford 大学にて APL System\7090 を試作
1966	IBM T. J. Watson 研にて APL\360 実験システムが稼動
1968	APL\360 (OS および DOS), APL\1130 完成 Marquardt-Commercial 社が最初の APL TSS サービスを開始.
1969	第1回 International APL Users' Conference を New York 州 Binghamton で開催
1970	I. P. Sharp がファイル・サブシステムを，R. H. Lathwell が共用変数を考案
1971	Scientific Time Sharing社がAPL＊PLUS File Subsystem を発表
1973	IBM システム/370 用の APLSV (APL Shared Variable) を発表
1975	IBM システム/370 用の VS APL を発表
1978～79	富士通，日立がMシリーズ用 APL を発表
1981	第12回 International APL Users' Conference を San Francisco で開催

† APL by Toru TAKESHITA (Software Development Center, IBM Japan, Ltd.).

†† 日本アイ・ビー・エム(株) ソフトウェア・センター

(a) 実際の応用：いくつかの分野で応用された上で，改良されてきた．

(b) 実行可能性：計算機による解釈・実行を可能にするため，計算機の機能上の制約を配慮している．

Yale 大学の名誉教授である Allan J. Perlis は，APL の長所として次の3つを上げている．

(a) 簡潔さ：複雑なことが単純に記述できる．

(b) 融通性：数多くの表現方法がある．

(c) 構成能力：自然語の散文のような芸術的可能性を持つ．

APL は，汎用のプログラム言語であるが，その主な特徴は次のようなものである．

言語の基本的対象 (primitive object) は，**配列** (スカラ，ベクトル，行列，3次元以上のもの) であり，配列指向の言語と呼ばれることがある．たとえば，$A+B$は任意の (同じ形の) 配列A，Bに関して意味を持つ．配列の形はρという原始関数で求まり，$A[2\ 3;\ 4]$ のように配列に指標 (index) を付けて，その特定要素 (単数または複数) を示すことができる．

演算に使われる記号は46種あるが，構文はごく単純である．文の型 (type) は，指定 (specification) (例：DIST←SPEED×TIME)，分岐 (branch) (例：→LOOP) とその他 (空文，注釈文，関数の見出しなど) の3種である．関数には優先順位がないので，かっこがなければ，右から左へ式の演算が行われる．**原始関数** (システムに初めから組み込まれている関数) の引数は1個 (右側に置かれる) か2個 (両側に置かれる) であり，**ユーザ定義関数**では引数なしの場合もある．なお，APL では，+，−，×，÷ など言語に固有の演算記号で表し，スカラを引数とする演算に使われるものを**原始スカラ関数**と呼んでいる．

次に，**語義論** (semantics) 上の規則も少ない．原始関数の定義は対象となるデータの表現に独立である．スカラとして定義された関数は，いずれもほかの配列に拡張され，対応する項目ごとに関数が適用される．原始関数には副作用がない．

プログラム中の文の実行順序の制御は単純である．右矢印を含む分岐文によりあらゆる型 (無条件，条件，計算など) の分岐 (例：→5は無条件に5へ飛ぶ，→(x>0/s) は $x>0$ のときに s へ飛ぶ) を表現できる．どのユーザ定義関数もその実行が終了すると，制御はそれを呼び出したところへ戻る．

原始関数を体系的に修飾する作用子があって，原始関数の使い道は大幅に拡大される．たとえば，／で表される**簡約** (reduction) は，並び (list) のすべての要素に掛かるように，関数を修飾する．一例として，+/L，×/L，⌈/L，⌊/L で，Lの全要素の和，積，最大，最小が求められる．

原始関数には，前述のスカラ関数のほかに，**複合関数**があり，配列の形態のデータの加工や出力の書式の指定に便利である．

以上のことから，表の形のデータを取り扱うことの多い経営事務の分野の適用業務のプログラミングがかなり容易になる．

原始関数には，単純な加減乗除から分類や書式の指定まであるが，いずれも1字の記号で表される．APL で使われる**特殊文字**は次の通りである．

(a) 算術：+ − × ÷ ⋆ ⍟ ○ | ⌊ ⌈ ! ⌹

(b) 論理と比較：∨ ∧ ⍱ ⍲ ~ < ≤ = ≥ > ≠

(c) 選択と構造：／ ＼ ⌿ ⍀ [;] ↑ → ⍴，⍎ ⍕ ⊖

(d) その他：∈ ⍳ ? ⊥ ⊤ ⍒ ⍋ ⌽ ⍉

APL で書かれた文を解釈・実行する処理系の特徴としては次のことがあげられる．

(a) 解釈方式で実行：データの大きさや形や型をユーザが定義しなくともよく，実行時に動的に決められる．

(b) 対話式でシステムとやり取り：プログラムの開発と実行のための操作が効率的である．

(c) 作業空間の利用：プログラムとデータがおかれるスペース (記憶装置内の区域) を作業空間といい，計算帳のように使われる．そのコピーを補助記憶装置に入れておいて，後でその名前で取り出す．

(d) ライブラリの使用：補助記憶装置内に設けられた領域に作業空間 (のコピー) を保管しておく．私用ライブラリと共用ライブラリがある．

FORTRAN, BASIC, COBOL, PL/I などとの差異には次のようなことが含まれる．

- 数学的記号の使用：英語の単語の代わりに単一文字の記号を採用，演算に使われる記号は46種あり，特殊な鍵盤を使用．
- 配列が基本的対象：作用子により関数からほかの関数を派生，分岐やループが少ない (約10分の1)．
- 文法的規則が少ない：構文は3種類，演算に優先順位がなく，右から左へ実行される．
- ハードウェアや OS から全く独立：データの属性の指定 (宣言) が不要．(端末と作業空間との間の) 入出力にファイルの OPEN，CLOSE が不要．
- 当初から対話式言語として設計：解釈方式で実

行，追加や修正が容易．

• プログラムが小さくてすむ：行数は他言語の1/5から1/10.

• プログラマの生産性が高い：バッチ方式でのFORTRAN, COBOL, PL/I と比較して，大幅に向上する．

• PL/I の機能で，APL で直接カバーされてないものは，ON 条件と異種のデータの混在などである．

APL ではプログラムが短かくてすむという例としては，40行位で計算機のモデル化(内部論理の記述)やアセンブラの作成ができることが報告されている．

APL 言語の利用は，1970年頃から特に IBM 社内で急速に進み，1975年以降に一般にも広がり始めた．日本では1976年にはわずか数社でしか APL の利用は見られなかったが，1978年には国産メーカが APL を発表し，そのころより IBM のユーザから普及し始め，IBM の中型以上のユーザ企業ではすでに2〜3割に達している．大学関係では，東大，京大，阪大，北大を始め，いくつかの大学で APL 処理系が利用可能になっている．

APL の適用業務としては，財務管理，人事管理，市場調査，予測，設計，製品管理，販売管理，コスト管理，目標管理，統計計算，シミュレーション，経営情報などの分析，統計，数値計算，グラフ化，大規模適用業務のプロトタイプ作成等々広範囲に亘っている．科学技術計算も少なくないが，大半は経営事務関連の分野である．データの分析，短時間に処理結果が必要な仕事，パーソナルな問題解決，定義が不完全で変化する仕事，大規模な適用業務の試作，CPU の使用が少ないデータ処理業務に最適である．

APL の利用の効果の例として，1980年4月に日本アイ・ビー・エム(株)が開催したエンド・ユーザ・サービス・シンポジュウムで発表された客先の事例15件の中で，1本のプログラムによる月間の人員の節約は5〜9人が1件，3〜4人が7件，1〜2人が7件あった．

3. 今後の見通し

当初は数学的な言語で科学技術計算用と見なされ，また専用システムに限られていた APL が，共用変数の概念の導入により，ほかのユーザや OS のサブシステムとのデータのやり取りが可能になり，さらに汎用の対話式システムの下で利用ができ，かつデータベース管理システムとの結び付きもできてきたので，その利用は単なるパーソナルな使い方から，企業内の主要なオンラインのオペレーショナル・システムの開発に利用されるようになっている．

APL の価値は，機能や価格性能比は言うまでもなく，使いやすさと特にプログラマの生産性によって，評価される．初心者にとっては，数学的記号や配列演算に抵抗を感ずる人は少なくないが，慣れるにつれて，その豊富な表現能力，融通性と拡張性と，テストのしやすさと修正の容易なこと，したがって，大きな適用業務の開発と保守の生産性が大幅に上昇することが理解される．

1970年代には，APL がより多くのシステム環境で使え，導入がしやすくなったのであるが，1980年代には標準的基本ソフトウェアとの整合性を高め，どのサブシステムの下でも，同じように使えることが期待されている．現在ユーザから要求されていることは，エラーの取り扱い，報告書の書式機能，文字データの分類などの導入，ファイル入出力の効率化である．バッチのジョブにも APL を使いたいという向きもあり，コンパイラの是非も論じられている．バッチ処理は現在でも (VM/CMS や TSO で) 不可能ではないが，誤りの発生に対する例外処理がむつかしい．

APL 言語の拡張については，文献[19),20)]に触れられているが，関数の引数として集合を考えるとか作用子の適用をユーザ定義関数や合成関数 (composition)，導関数 (derivative) 等への拡張などがある．これらや複素数や一般的配列 (general array，配列の要素が配列であるもの，COBOL でいうレコードも取り扱う) の演算も IBM の T. J. Watson 研究所における実験的システム APL 2 に組み込まれている．

なお，APL 言語の標準化は，1979年3月に ISO/TC 97 がその活動開始を承認しているが，まだ本格的には進められていない．規格の草案は，1979年5月に New York 州 Rochester で開催された APL Conference '79 で提案されている．また，ユーザ間の作業空間 (APL プログラム) の交換をしやすくするため，その標準化も ACM の STAPL (Special Technical Committee for APL of the ACM) で検討されており，その規格案は STAPL の機関誌 APL Quote Quad (Vol. 8, No. 2, 1978) に発表されている．

4. 例 題

APL の特徴の一端を表すような例題を以下にあげておこう．

例1 2次方程式 $y=ax^2+bx+c$ の根を，$a\neq 0$

で，$b^2-4ac \geqq 0$ のときに，求める関数（サブルーチン）を定義する．答えの出力は，フィールドの長さ10，小数点以下のけた数を5けたとする．

```
    ∇QUAD
[1] D←(B*2)-4×A×C
[2] →((0=A)∨(0≥D))/0
[3] X←((-B)+((1 ¯1)×D*0.5))÷2×A
[4] 10 5⍕X∇
```

2行めと3行めは1行にできる．これをFORTRANで書くと，11行を要する．∇は関数定義の始めと終わりを示す．

例2　平均，標準偏差，相関係数を求めるプログラムを作成してみる．XとYは同じ長さのベクトルで，その要素はあらかじめ与えられているとする．

```
    ∇STAT
[1] XX←X-MX←(+/X)÷ρX
[2] YY←Y-MY←(+/Y)÷ρY
[3] SX←((+/XX*2)÷ρX)*·5
[4] SY←(+/YY*2)÷ρY)*·5
[5] XX←XX÷SX
[6] YY←YY÷SY
[7] R←(+/XX×YY)÷ρX∇
```

+/XはXの要素の和を，ρXはXの要素の個数を求める．

例3　5行3列の数値データMがあるとき，横書きの氏名NMを左側におき，それぞれフィールドの長さと小数点以下のけた数を7と3，5と2，7と3に取って，3つの数値を氏名の右側におき，かつ見出しHと1行分のスペースをあけるには，次のようにする．⍕は数値Mを左側の引数で指定されたように文字化する．

```
    CHM←7 3 5 2 7 3⍕M
    H,[1]' ',[1]NM,CHM
  SAMPLE  REPORT
MURAI 36.832 8.26 39.231
OKA   29.635 3.15 66.382
SAKAI  5.370 0.23 34.342
YASUI 23.631 3.61 58.321
WAKO   4.023 8.32  6.581
```

参考文献

1) Iverson, K. E.: A Programming Language, p. 286, John Wiley & Sons, New York (1962).
2) 長田純一，内山　昭：APLプログラミング言語（上記の邦訳），p. 362，講談社 (1975).
3) Falkoff, A. D., Iverson, K. E. and Sussenguth, E. H.: Formal Description of System/360, IBM Systems Journal, Vol. 3, Nos. 2 and 3, pp. 198-262 (1964).
4) Iverson, K. E.: Elementary Functions: An Algorithmic Treatment, Science Research Ass. Inc. Chicago (1966).
5) 和田　弘：アルゴリズムによる初等関数（上記の邦訳），p. 243，共立出版 (1969).
6) Iverson, K. E.: ALGEBRA: An Algorithmic Treatment, p. 361, Addison-Wesley Publishing Co. (1972).
7) Falkoff, A. D. and Iverson, K. E.: The Design of APL, IBM Journal of Research and Development, Vol. 17, No. 4, pp. 324-334 (1973).
8) Lathwell, R. H.: System Formulation and APL Shared Variables, IBM Journal of Research and Development, Vol. 17, No. 4, pp. 353-359 (1973).
9) Falkoff, A. D. and Iverson, K. E.: The Evolution of APL, Proceedings of A Conference on the History of Programming Languages, ACM SIGPLAN (1979).
10) APL Language, GC 26-3847-3, p. 100, IBM Corp. (1978).
11) APL 言語 N:GC 26-3847 （上記の邦訳），p. 108，日本アイ・ビー・エム (1978).
12) APL 入門テキストブック，N:GR 18-5007, p. 170，日本アイ・ビー・エム (1978).
13) Iverson, K. E.: An Introduction to APL for Scientists and Engineers, G 320-6178-0, p. 27, IBM Corp. (1978).
14) Gilman, L. and Rose, A. J.: APL - An Interactive Approach, Second Edition Revised Printing, p. 384, John Wiley & Sons. Inc. New York (1974).
15) Polivka, R. P. and Pakin, S.: APL: The Language and Its Usage, p. 579, Prentice-Hall Inc. N. J. (1975).
16) Harms E. and Zabinski, M. P.: Introduction to APL and Computer Programming, p. 400, John Wiley & Sons, Inc. N. Y. (1977).
17) The APL Handbook of Techniques, S 320-5996-0, p. 112, IBM Corp. (1978).
18) APL Programming Guide, G 3201-6103-0. p. 101, IBM Corp. (1978).
19) Iverson, K. E. (訳　竹下　亨)：作用子 (Operators)，情報処理，Vol. 20, No. 9, pp. 766-778 (1979).
20) Iverson, K. E.: Notation as a Tool of Thought, 1979 ACM Turing Award Lecture, Comm. of the ACM, Aug. 1980, Vol. 23. No. 8, pp. 444-464 (1980). 邦訳は「bit」 Vol. 13 No. 6～8 共立出版 (1981).

21) 長田純一，内山 昭：APLSV. p. 342，丸善 (1975).
22) 長田純一，内山 昭：APL 入門，p. 243，丸善 (1976).
23) 竹下 亨：APL，情報処理，Vol. 19, No. 1, pp. 78-85 (1978).
24) 竹下 亨：APL プログラミング入門，p. 206，オーム社 (1978).
25) 竹下 亨：プログラミング言語 APL，p. 232，共立出版 (1981).
26) Orth, D. L：CALCULUS in a New Way, APL Press, Pa. (1976).
27) 日本 APL 協会編訳：APL-微積分のアルゴリズム的方法(上記の邦訳)，p. 320，工学図書(1980).

(昭和56年1月6日受付)