

[03_06]九州大学大型計算機センター広報 : 3(6)

<https://doi.org/10.15017/1467971>

出版情報 : 九州大学大型計算機センター広報. 3 (6), pp.1-70, 1970-12-18. 九州大学大型計算機センター
バージョン :
権利関係 :

FORTRANを基礎にしたプログラム言語について(II)

牛 島 和 夫^{*}

1. DYSTALについて

今回は、FORTRANを基礎にしたプログラム言語の一例として、DYSTAL (DYnamic STorage ALlocation program in FORTRAN)を紹介する。DYSTALは、その製作者 J. M. Sakoda によって1966年イタリアのピサで開かれたIFIP (国際情報処理連合) の記号処理に関する国際会議で報告された [1]。筆者はたまたまそのマニュアル [2] を手に入れることができたので、ある種の記号処理を行うための道具として、九州大学大型計算機センターのFACOM 230-60 FORTRANでimplementし、先頃ライブラリーに登録した。1963年に発表されたSLIPはかなり多くの利用者がおり、SLIPを使って、記号処理、数式処理や言語分析などを行った例はいくつか発表されているが、ここで紹介するDYSTALについては、必ずしも広く使われてはいないようである。実際PL/Iが真の意味で利用可能になれば、DYSTALがわざわざ登場する必要もないものと思われる。それにもかゝらず、ここでDYSTALをとりあげることにしたのは、次のような理由による。

1) DYSTALがFORTRANを基礎にして言語を拡張する際の基本的な点を非常に多く含んでおり、FORTRAN語の特質を見事にとらえている。

2) データ構造とその処理をFORTRANの側面からとらえてみる。

3) すべての手続きがFORTRANで記述してある。

また、センター利用の立場から考えて、

4) 九大のセンターは当分FORTRANとALGOLのジョブが中心で、PL/Iが使用できるようになっても、全体的なサービス体制、言語システムの安定性からFORTRANならば安心できる。

2. 記憶場所とその内容

DYSTALを一口でいえば、「特に動的な記憶領域の使用を可能にするための多くの関数といくつかのサブルーチンの集合」といえる。FORTRANでは、使用者の立場からいえば、記憶の内容と記憶の場所の概念をあまりはっきり分離させていない。普通には、変数名は計算機の側からいえば記憶場所であり、プログラムを書く側からは、記憶の内容である値に意味がある場合が多い。しかしある変数とある変数の場所と場所の相互関係はソースプログラムからは不明である。もっとも逆にいえばFORTRANのような手続き向き言語が開発されたのは、機械語のプログラミングにおける記憶場

* 九州大学工学部通信工学科

所の割付けの繁雑さからの解放でもあったわけである。もちろん、FORTRANでも、COMMONやEQUIVALENCEを駆使することによって場所と内容の対応はある程度つけることはできる。もっとも普通の数値計算を行っている分には、特に場所と内容の対応が必要になることは少ない。

番地と内容の対応づけをDYSTALでは、次のような方法で擬似的に実現している。すなわち、共通領域に大きく一括してとった配列LOT（またはFLOT）の内部でLOT（1）を基準にして相対的に番地づけを行う。LOT領域の中には、いろいろの型をもったデータが存在し得る。DYSTALでは、標準的には単精度（1語長）のデータだけをあつかいそれらの型は表1のようになっている。ここでModeはそれぞれの型に割付けられた符号である。詳細は後述するが、整数、文字型データに対してはLOT（・）で実数型データはFLOT（・）で処理する。

Mode	データの型	入出力FORMAT
1	リストの名	I 5
2	整数	I 10
3	F型実数	F13.5
4	文字型データ(1語単位)	A 4
5	文字型データ(テキスト)	17A 4
6	E型実数	E13.5

表 1

DYSTALを使った簡単なプログラムを図1に示す。これは行列AとBの要素を読みこんで $A^T \cdot B^{-1}$ の結

果を行列Cに与えるプログラムである。サブルチンINLOTはDYSTALシステムの初期設定、LSREADは、DYSTALシステムの標準入力プログラム。次の行は $C = A^T \cdot B^{-1}$ の操作。サブルチンKDUMPは標準出力プログラムである。IDUMMYは広報の前号の第7節に書いたダミー変数である。

```

C
COMMON LOT(2000)
DIMENSION FLCT(2000)
EQUIVALENCE (LOT,FLOT)
C
CALL INLOT(10,2000)
IDUMMY=LSREAD(N)
IDUMMY=MATMP(MTRAN(LOT(N+1)),MATINV(LOT(N+2)),LOT(N+3))
CALL KDUMP
STOP
END
    
```

図 1

3. DYSTALのリスト

図1のプログラムに見るように、共通変数LOTは1次元配列で計算機内で1次元的に並んだ番地にそのまま対応する。この領域を先頭（LOT（1））と末尾（ここではLOT（2000））の両端から必要に応じて一かたまりになったデータを取りあつかうための連続した領域を切り出してゆく。

この一つ一つの連続した領域のことをDYSTALでは**リスト**と呼んでいる。このようにしてLOT領域から、必要に応じてリストを切り出してくるとある時点で図2のようにになっているであろう。そしてLOT領域の中に使用領域と未使用領域ができる。このようなLOT領域の監視と制御が、DYSTATの関数群によって行なわれるわけである。ここでLOT(1)の側からはじまるリストを**永久領域 (parmanent area)**のリスト、末尾からはじまるリストを**一時的領域 (temporary area)**のリストと呼んでいる。ここでLIST1とLIST2は特別な意味をもったリストで、サブチンINLOTでDYSTATシステムの初期設定を行う際に作成され、LIST1は主として上述の使用領域と未使用領域の監視のためのデータ、LIST2は表1にあげた標準的なModeのデータの入出力のためのFORMATが格納されている。したがって利用者が実際に利用するのはLIST3以下の領域になる。

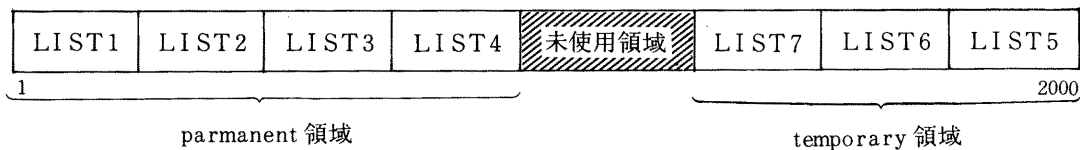


図2 LOT領域

次にリストの内部構造をしらべてみよう。リストは**頭部 (head)**と**本体 (body)**と呼ばれる2つの部分からなっている。headは5記憶単位からなっていてリストの区別、性質、大きさなどを示すために用いられる。それらはIden, Node, Mode, Nmax, Nctr (Ncounter)と名づけられている。

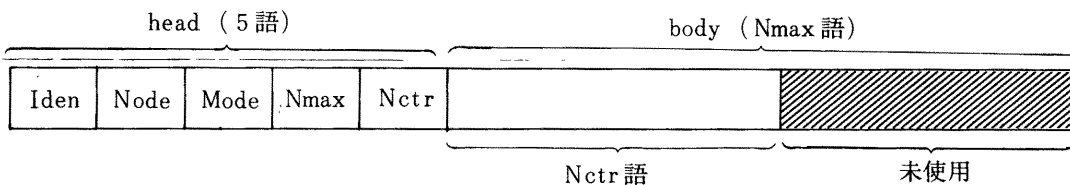


図3 リスト

Idenにはこのリストを区別するための4文字以内の文字型データのラベル。

Nodeについては後述する（通常は0が入る）。

Modeはリストの本体に入るデータの型に応じた表1の1～6の符号。

Nmaxはリスト中の最大データ収納語数。

NctrはNmax語中実際に使用している語数。

表1にあげたMode = 1、**リストの名**は、LOT領域の中で当該のリストのNctrのセルが占める場所の番地（LOT(1)の場所を1としたLOTの添字）が与えられる。したがってリストの名は、LOT領域の中のリストの存在場所と等価である。このようにリストの名をきめると、たとえば、LISTA = 1000が与えられていると、リストLISTAのMode, Nmax, Nctrの値は

LOT (LISTA - 2)

LOT (LISTA - 1)

LOT (LISTA)

によってそれぞれとりだすことができる。

いま、 $LOT(LISTA - 2) = 2$ （整数）であれば、リスト中の第1番目のデータ、第2番目のデータ、……は、

LOT (LISTA + 1), LOT (LIST + 2), …

としてとりだすことができる。

このリストの使用部分のすぐ次(Nctr + 1)に新しくデータIWDを追加するには、

IX = LOT (LISTA) + 1

LOT (IX) = IWD

LOT (LISTA) = IX ……………Nctrを1あげる。

とすればよい。しかしこのような必要が生ずるごとに上のような文を書いていたのでは、繁雑なのでこれを関数にまとめて

FUNCTION LOAD (IWD, LIST)

を定義する。IWDは付加したいデータ、LISTはリストの名である。また関数値はIWDをそのまま返す。このようにすると利用者はリストの中に現在どれだけデータがたまっているかに関心を払うことなく、リストをあたかもpush down stack（前号第3節参照）のように使うことができる。したがって関数LOADはpush downの操作に対応している。もちろんNmax < Nctrとなるような操作に対しては警告が発せられる。push downに対して、Nctr番目のデータをとりだしてくるpop upの操作を行う関数

FUNCTION ITAKE (LIST)

も用意されている。LISTはリストの名で、関数値はpop upされた値である。ITAKEの操作

をFORTRANの基本語彙で記述すれば

```
JTEMP=LOT (LIST) +LIST
```

```
ITAKE=LOT (JTEMP)
```

```
LOT (LIST) =LOT (LIST) -1 .....Nctrを1さげる
```

ところで、LOT (LIST) =3 (実数) であることがわかっているとき

```
A=ITAKE (LIST) *B
```

とすると、実際に関数値として返されたデータの内容は実数であるにもかかわらず、ITAKEが整数型の関数であるために、FACOM 230-60 FORTRANでは異った型の演算のため自動的に換算関数が挿入されてしまうのでAに正しい値が与えられない (JISFORTRANでは異なった型の演算は許されない)。だからといって

```
A=FLOAT (ITAKE (LIST) ) *B
```

とやっても同じようなことになる。そこで実数の内容が換算されないでそのまま受け渡されるような関数

```
FUNCTION FLT (IWD)
```

```
EQUIVALENCE (I, A)
```

```
I=IWD
```

```
FLT=A
```

```
RETURN
```

```
END
```

が用意され

```
A=FLT (ITAKE (LIST) ) *B
```

とする。このようなことは、FORTRANで記号処理を行なおうとする場合にはよく生じ、SLIPでも同じようなテクニックを用いている (SLIPでは関数INTEGER)。

LOADおよびITAKEは、リストをpush down stackのようにあつかう関数であったがリストの第I番目のデータの内容をとりだすためには、

```
IX=LIST+I
```

```
IWD=LOT (IX)
```

とする (FACOM 230-60 FORTRANでは、LOT (LIST+I) という用法が許されるが、一般には配列の添字はc, c' を整数型定数、jを整数型変数としてc * j ± c' という型式だけが許されている。互換性の上から、ここではこちらの用法にしたがう)。ここでもリストのI番目のデータをとりだすための関数を用意して

```
FUNCTION ITEM (I, LIST)
```

関数値は、I 番目のデータである。このリストが実数型データを格納している場合のために

FUNCTION FITEM (I, LIST)

も用意されている。

$LOT(LISTA-2) = 2$, $LOT(LISTB-2) = 3$ のとき

N = ITEM (LOT (LISTA), LISTA)

S = FITEM (LOT (LISTB-1) - 1, LISTB)

によって N, S に代入された値は、それぞれのリストの中のどんなデータであるか考えてみるとよい。

次に ITEM の逆操作、すなわちリストの I 番目にデータを格納するために

FUNCTION IPLACE (IWD, LOCWD)

がある。関数値は IWD。変数 IWD の内容をリストの名 LIST の I 番目に格納するためには、

IDUMMY = IPLACE (IWD, LIST+I)

とする。またリストの I 番目の内容と J 番目の内容の積を K 番目に格納する操作は次のように書ける

IDUMMY = IPLACE (ITEM (I, LIST) * ITEM (J, LIST), LIST+K)

関数値を IWD にしておくことと便利なことは、次の用法で知れよう。上の積をさらに K+L 番目にも格納したいという場合に

**IDUMMY = IPLACE (IPLACE (ITEM (I, LIST) * ITEM (J, LIST), LIST+K),
LIST+K+L)**

とすればよい。これは一般の FORTRAN では許されていない多重代入文 (ALGOL でいえば $A := B := C := 1$ という用法) を可能にさせている苦肉の策でもあるわけである。したがって上の文を形式的にかけば

LOT (LIST+K) = LOT (LIST+K+L) = LOT (LIST+I) * LOT (LIST+J)

と同じことである。一般の FORTRAN では、先に述べたように添字表現は制限されているけれども、関数の引数には、式が自由に書ける利点を利用しているものといえよう。

なお IPLACE と同じような関数であるが

FUNCTION ISTORE (IWD, IVAR)

が用意されている。これは、代入の際の換算をきらい向きのための関数で

IVAR = IWD

と同じような意味をもっている。

以下は内輪話になるがこれには FORTRAN コンパイラの特長事情がかくされている。すなわち広報前号第 7 節に述べたように副プログラムの仮引数と実引数の対応について、JIS では、順序、個数および型が一致しなければならないと述べている (JIS p.34、p.36)。ところが、FORTRAN はプログラム単位がそれぞれ独立しているので、コンパイル時に、実引数と仮引数の型の対応を調べることは原理的に不可能である。ここにも、パラメタの引き渡しの

際にパラメタの場所を引き渡さなければならない事情がある。そこで結局、J I Sで規定しているように実引数と、仮引数の型の一致をしらべるのは、目的プログラムの実行時しかないわけである。ところが実行時にひとつひとつ型の一致をしらべるような目的プログラムを生成していたのでは、これは実行速度が遅くなってしまって現実的でない。このため現存するFORTRANプロセッサで現実に型の一致をしらべているものは一つもないといってよいだろう（少なくとも我国の大学の大型計算機センターの3つの機種種のFORTRAN処理系はしらべていない）。ではJ I Sではできもしないことをなぜ規定にするかといえば、J I Sはコンパイラ製作者だけに対する規定ではなく、実際にFORTRANでプログラムを書く人にとっても一つの基準であると考えべきで、少くとも型が一致するようなプログラムを書いてさえおけば、プログラムは正しく実行されることを保証すべきであることを主張していると考えるのが妥当である。そこで本論にもどるとたとえば、変数Aで示される記憶場所に入っている内容（文字型の内容であるとする）を整数Iで示す記憶場所に移すためには

$$I = A$$

という文では、自動的に換算関数が挿入されてしまって、FACOM 230-60では、最後の9ビット（文字型でいえば4文字目）が主として変化を受けて、意図したデータの転送ができなくなる。ところがこれを

$$I \text{ DUMMY} = I \text{ STORE} (A, I)$$

とすると、幸いなことに(?)副プログラムに引き渡されるのはAとIの記憶番地であって、Aが実、Iが整という情報は引き渡されないので、副プログラムの中では(I VAR = I WD)という文が実行され、データの換算もなく意図通りのデータの転送を行なうことができる。普通の数値計算ではこのような一種の芸当をすることはほとんど不要であるし、あえてこのようなことはしない方がよいが、FORTRANで文字処理などを行おうとすると、このような点が面倒な点である。

4. リストの生成と削除

LOT領域にリストを生成するために

$$\text{FUNCTION LSTALL (MOD, MAX, LIST)}$$

が用意されている。MODは作成したいリストのMode (1~6), MAXはNmaxの値、LISTは、生成されたリストの名が与えられる。関数値はLISTが返される。生成されたリストの本体は帰零されていないので、もし0であることが望まなければ

$$\text{FUNCTION ICLEAR (LIST)}$$

を用いるとLOT (LIST+1), …… , LOT (LIST+Nmax) の内容が全て0にされる、関数値はLIST。またLSTALLによって生成されたリストのI denセルには、このままではラベルが入っていない。もしラベルが必要ならば

$$\text{FUNCTION LABEL (LIST, IDEN)}$$

によって、LISTのI denセルにラベルを代入すればよい。関数値はLIST。いまLISTNを

新しく生成して、内容を帰零し、I denセルに▼NAME▼というラベルをつけるには、

```
      I DUMMY=LABEL (ICLEAR (LSTALL (1, MAX, LISTN) ), 4HNAME)
```

この場合、このリストLISTNは permanent 領域に生成される。もし temporary 領域に生成したければ、そのためのスイッチをONにしなければならない。この操作をするのが、

```
      FUNCTION LTEMP (MAX)
```

である。ここでMAXはダミーである。なんとなれば、FORTRANの関数は必ず1つ以上の引数をもたなければならない。そうであれば、LTEMPはサブチンでもよいのだが、Sakoda は、関数の重疊的使用をとことんまで行うために、LTEMPの関数値はMAXをそのまま返すことにして慣用的に、

```
      I DUMMY=LSTALL (2, LTEMP(100) , LISTA)
```

のように、LSTALLの第2引数にかぶせるように使っている。LSTALLが呼ばれると、スイッチは自動的にOFFになる。

例 temporary 領域に、実数を取扱うリスト (Nmax = 10) を15個生成し、各々のリストの名をリストLISTNに生成順にたくわえる。なおLISTNは permanent 領域に生成する。

```
      I DUMMY=LABEL (LSTALL (1, 15, LISTN) , 4HLSTN)
```

```
      DO 1 I = 1, 15
```

```
      I DUMMY=LOAD (LSTALL (3, LTEMP (10) , NOM) , LISTN)
```

```
1 CONTINUE
```

次に不要になったリストをLOT領域から削除するために2つの関数

```
      FUNCTION LERASE (LISTA)
```

```
      FUNCTION KERASE (LISTA)
```

が用意されている。関数値はいずれもLISTA。前節で述べたように、DYSTALではLOT領域を先頭と末尾から別々につめて使用しているので、もっとも新しく作られたリストを削除したい場合(例えば図2のLIST4, LIST7)には、問題はないが、図2のLIST3やLIST5を削除したい場合は面倒なことがおこる。LERASE(LIST2)とすると、LIST2, LIST3, LIST4が、LERASE(LIST6)とするとLIST7, LIST6 がそれぞれ同時に削除される。一方KERASEでは、KERASE(LIST4)のようにはじめのリストの場合はそのまま削除されるが、KERASE(LIST5)のような場合は、削除可という印をつけておいて、LIST7, LIST6が削除されたときに同時に削除するような苦心を払っている。

5. リストの標準入出力

リストの標準的な入力のために

FUNCTION LSREAD (NAME)

が用意されている。次のような図4の4つのリストLISTA, LISTB, LISTC, TEXTを、LOT領域に読みこむためには、例えば図5のようにカードを用意すればよい。

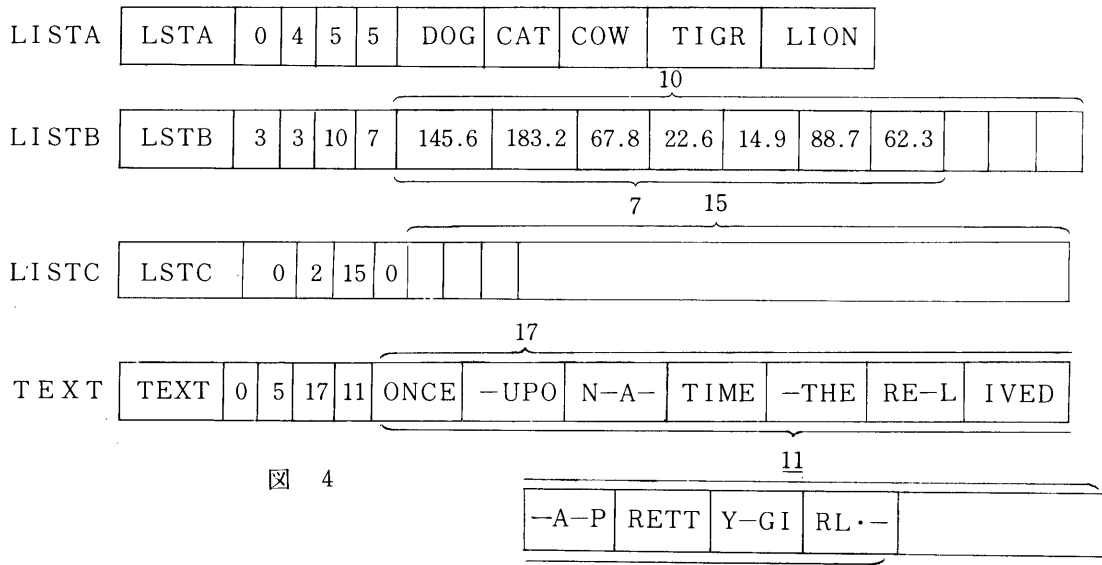


図 4

Iden	Node	Mode	Nmax	Nctr						
2	8	4	16	28	30	42	44	56	58	70
	NAME					1		10		0
2	LSTB		3			3		10		7
	145.6		183.2			67.8		22.6		14.9
	88.7		62.8							
1	LSTA		0			4		5		5
	DOG		CAT							TIGR
										LION
3	LSTC		0			2		15		0
4	TEXT		0			5		17		11
	ONCE UPON A TIME THERE LIVED A PRETTY GIRL.									
	STOP									

図 5

図5の説明（1枚のカードは各14欄づつ5コマに分割する）。

1枚目のカード…後述。

2枚目…LISTBのheadの情報を記述したカード。8欄目の2はリストの番号である。

3～4枚目…LISTBのNctrセルの7に対応するMode=3（F型実数）のデータ7個。

5枚目…LISTAのheadの情報、8欄目の1はリストの番号。

6枚目…Mode=4, Nctr=5に対応する文字型データ5個。

7枚目…LISTCのheadの情報、8欄目の3はリストの番号、Nctr=0だからデータのカードはない。

8枚目…TEXTのheadの情報、8欄目の4はリストの番号

9枚目…textの内容。Mode4とMode5のちがいは、前者は1語1語の4文字が独立したイミをもっているが、Mode5はtext全体として意味をもっており、各語はそのtextを便宜的に4文字ずつ区切った情報が入る。

10枚目…入力カード終の制御カード。

さて1枚目のリストNAMEは、リストの名をエンタリーとするリストである（したがってMode=1）。これは、2枚目以下で読みこまれたリストが、LOT領域に生成されて各々のリストの名が確立するとそのリストの名を8欄目のリストの番号1, 2, 3, 4に応じて、

LOT (NAME+1), LOT (NAME+2), …

に格納するindexの役割をもったリストである。LSREADの引数NAMEは、このリストNAMEのリストの名が与えられて出力される。またLSREADの関数値はリストの番号が1のリストの名すなわちLOT (NAME+1)の内容が返される。なお図4または図5の2番目のリストLISTBのNode=3は、このようにしてLOT領域に確定した3番目のリストの名（ここではLISTCの値）を最終的にこのNodeセルに代入することを意味している。このようにして、DYSTALでは、特別の関係にある2つのリストを結合する手段を与え、これを**結節リスト** (Nodal list)と呼んでいる。ところで図5のように入力カードを用意すると、これらのリストは全てpermanent領域に生成されるが、temporary領域に生成したい場合は、リストのheadの情報を記述するカードの第2欄に1をパンチしておけばよい。

図 6

さてこれらの標準入力のためのFORMAT	Iden Node Mode Nmax Nctr	4 × 4 文字
は先に述べたように、図3の2番目のリストに	LFMT	5. 24. 24. (5(9X, I5))
格納されている(図6)。Mode iに対応する		(5(4X, I10))
FORMATは、LOT (LFMT+4*(i		(5(1X, F13.5))
-1)+1)~LOT (LFMT+4*(i-1)		(5(10X, A4))
+4)に入っている		(1X, 17A4)
ので、LSREADが補助		(5(1X, E13.5))

的に呼んでいる関数

```
FUNCTION IREAD (MYFMT, I, N, LISTA)
```

を

```
IFMT=LFMT+4*(LOT(LISTA)-1)+1
```

```
IDUMMY=IREAD(LOT(IFMT), 1, LOT(LISTA), LISTA)
```

としてやればよい。ここでMYFMTは配列名、LISTAはリストの名でリストの第I番目から第N番目の要素までを読みこむ。関数の中では、例えば

```
READ(5, MYFMT) (LOT(J), J=1, N)
```

のような文が実行されることになる（入出力文のFORMATの指定は文番号ではなく配列名でもよいことに注意）。

次に標準的な出力用には

```
SUBROUTINE KDUMP
```

```
FUNCTION IDUMP(LISTA, LISTB)
```

が用意されている。サブルーチンKDUMPは、LOT領域中のすべてのリストを同じ図6のFORMATを使って出力する。図4をそのままKDUMPで出力した例を図7にしめす。

図7 計算機の出力

```
DUMP OF LIST 1 NAME 5 IDEN = INLO NODE = 0 MODE = 2 NMAX = 25 NCTR = 25
      146      2000      2000      0      35
      0      0      0      0      0
      0      0      0      0      0
      0      0      0      0      0
      0      0      0      0      0

DUMP OF LIST 2 NAME 35 IDEN = LFMT NODE = 0 MODF = 5 NMAX = 24 NCTR = 24
(>(9X,15)) (5(4X,110)) (5(1X,F13.5)) (5(10X,A4)) (1
X*17A4) (5(1X*E13.6))

DUMP OF LIST 3 NAME 64 IDEN = NAME NODE = 0 MODE = 1 NMAX = 10 NCTR = 4
      94      79      104      124

DUMP OF LIST 4 NAME 79 IDEN = LSTB NODE = 104 MODE = 3 NMAX = 10 NCTR = 7
      145.60000      183.20000      67.80000      22.60000      14.90000
      88.70000      62.80000

DUMP OF LIST 5 NAME 94 IDEN = LSTA NODF = 0 MODE = 4 NMAX = 5 NCTR = 5
      DOG      CAT      COW      TIGR      LION

DUMP OF LIST 6 NAME 104 IDEN = LSTC NODE = 0 MODE = 2 NMAX = 15 NCTR = 0

DUMP OF LIST 7 NAME 124 IDEN = TEXT NODE = 0 MODE = 5 NMAX = 17 NCTR = 11
ONCE UPON A TIME THERE LIVED A PRETTY GIRL.
```

関数 IDUMP は、LOT 領域中の LISTA から LISTB までのリストを出力する。したがって、LISTA だけの出力をする場合は、

IDUMMY=IDUMP (LISTA, LISTA)

とすればよい。KDUMP が IDUMP を使用している。IDUMP は IREAD に対応する

FUNCTION IPRINT (MYFMT, I, N, , LISTA)

を使用している。

Variable FORMAT これまでは、リストの各要素はすべて同一の Mode であったが必ずしも同じである必要はない。そのようなリストのために可変 FORMAT を用意することができる。そのためには、図 6 の FORMAT に対応するような FORMAT 用の Mode 5 のリストを利用者があらかじめ作成しておいて、可変 FORMAT をもつリストの Mode セルに、対応する FORMAT をもつリストのリストの名をいれておく。このようにしておけば、入力または出力の際に

IFMT=LOT (LISTV-2)

IDUMMY=IREAD (LOT (IFMT+1), 1, LOT (LISTV), LISTV)

のような用法で実行できる。

6. 行列の取扱い

FORTRAN では、2 次元の配列

DIMENSION A (m, n)

は 1 次元的に A (1, 1), A (2, 1), …, A (m, 1), A (1, 2), A (2, 2), …, A (m, 2), …, A (1, n), A (2, n), …, A (m, n) のように列方向に割付けるように約束されている。この割付は、コンパイラが行っているわけであるが、DYSTAL では、LOT 領域の中に行列を格納する際に、実行時に行なわなければならない。3 × 4 の行列 $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$ は、図 8 のように行方向に格納する。

Iden	Node	Mode	Nmax	Nctr												
MATA	-4	2	12	12	1	2	3	4	5	6	7	8	9	10	11	12

図 8

ここで Node セルが利用されている。すなわち列の数に負号をつける。逆にいえば DYSTAL システムは Node セルが負であれば、そのリストが行列であることを知るわけである。

```

    IF (LOT (MATA-3)) 1, 2, 2
1  N=IABS (LOT (MATA-3))
    M=LOT (MATA-1) / N
    :
```

によって、列の数N、行の数Mを知ることができる。MATAの(I, J)要素をとり出すためには

```

    IJ=MATA+J+(I-1)*IABS (LOT (MATA-3))
    IWD=LOT (IJ)
```

そこで、MATAの第I行目を知るために、

```

FUNCTION LINE (I, MATA)
```

が用意されている。

```

    LINE=(I-1)*IABS (LOT (MATA-3))+MATA
```

この関数値は、実際にはA(I-1, N)の場所であるが、DYSTALのリストの慣用にしたがってわざわざLOT(LINE+1), LOT(LINE+2), ……がA(I, 1), A(I, 2), ……に対応するようにしてある。したがって(I, J)要素を取り出すには、

```

    IWD=ITEM (J, LINE (I, MATA))
```

(I, J)要素に格納するには

```

    IDUMMY=IPLACE (IWD, J+LINE (I, MATA))
```

とすればよい。行列のための標準入出力も、LSREAD、KDUMPに用意されている。

7. その他の基本的なDYSTAL関数

基本的なDYSTAL関数としてこのほかつぎのような関数が用意されている。

```

IDLETE (I, LISTA)
```

LISTAの第I番目の要素を削除する。I+1番目以下が順次くりあがり、Nctr=Nctr-1となる。関数値は削除された要素の値。

```

INSERT (IWD, I, LISTA)
```

値IWDをLISTAのI番目のセルに挿入する。旧I番目以下は1づつくりさがる。Nctr=Nctr+1。関数値はIWD。

```

LOCATE (IWD, LISTA, I)
```

LISTAのI番目の要素からはじめて、値IWDをもっている要素の番号を関数値とする。みつからない場合は0を返す。

```

ICOPY (LISTA)
```

LISTAと同じ内容のリストを生成する。関数値は新しいリストの名。

LCOPY (N, LISTA, LISTB)

LISTAのN個の要素をLISTBの余白に付加する。LISTBのNctr = Nctr + N。関数値はLISTB。

IMOVE (N, LISTA, LISTB)

LISTAのN個の要素を番号の小さい方から順にLISTBに移す。関数値はLISTA。

IRAISE (N, LISTA, LISTB)

LISTAのN個の要素を番号の大きい方から順にLISTBに移す、関数値はLISTA。

7節までに述べてきた基本的な関数群にもとづいて、問題向きの関数群がさらに数組用意してある。

すなわち

統計計算

行列計算

鎖構造の処理

木構造の処理

文字処理

分類、ませ合せ

なお、もともとのDYSMALシステムにある磁気テープ関係の関数は、九州大学大型計算機センターの事情を考慮して implement しなかった。

8. 簡単な統計計算と行列計算

Mode = 3 のデータをたくわえている LISTA の要素の平均値あるいは分散をもとめるには、たとえば、

```
SUMX = 0. 0
```

```
SUMXX = 0. 0
```

```
DO 2 0 I = 1, N
```

```
IX = LISTA + I
```

```
SUMX = SUMX + FLOT (IX)
```

```
SUMXX = SUMXX + FLOT (IX) ** 2
```

```
2 0 CONTINUE
```

```
FMEAN = SUMX / FLOAT (LOT (LISTA))
```

```
VAR = (SUMXX - FMEAN ** 2) / FLOAT (LOT (LISTA))
```

とすればよい、これらの処理を関数化して、次のような統計用の関数が用意されている。

- SUM (N, LISTA)** 実数の要素をN個加えて関数値とする。
- ISUM (N, LISTA)** 整数の " "
- SUMSQ (N, LISTA)** 実数の要素の各2乗のN個の和を関数値とする。
- ISUMSQ (N, LISTA)** 整数の " "
- SUMXY(N, LISTA, LISTB)** 実数の要素をもつ2つのリストのN個の積和を関数値とする。
- VAR (N, LISTA)** 実数の要素をもつリストのN個の分散を関数値とする。
- LADD (N, LISTA, LISTB, LISTC)** 実数の要素をもつ2つのリストLISTAとLISTBの対応する要素を加えてLISTCの対応する場所に格納する (N次元実数ベクトルの和)。関数値はLISTC。
- LADDI (N, LISTA, LISTB, LISTC)** N次元整数ベクトルの和、関数値はLISTC。
- LSUB (N, LISTA, LISTB, LISTC)** LISTBの要素からLISTAの要素を引く。関数値はLISTC。
- LSUBI (N, LISTA, LISTB, LISTC)** 同上、ただし整数のリスト。

行列計算用には次のような関数が用意されている。ここで、説明の便利のため、リストMATA, MATB, MATC, MATRなどにはそれぞれ行列A, B, C, Rの要素が都合よく格納されているものとする。

- MATMP (MATA, MATB, MATC)** $C = AB$, 関数値はMATC。
- MTRAN (MATA)** A^T , 関数値はMATA。
- MPTRAN (MATA, MATB, MATC)** $C = A \cdot B^T$, 関数値はMATC。
- MTRIAN (MATR)** 対角要素が全て正である対称行列Rの三角化。
- MATINV (MATA)** A^{-1} , 関数値はMATA

特にデータ構造と関連して興味のある鎖構造, 木構造, 文字処理の関数群については、次回に多少詳細に紹介することにする (以下次号)。

文献

- [1] D. G. Bobrow (ed.), Symbol Manipulation Languages and Techniques pp. 487 North Holland (1968), Sakodaの論文はこの本のpp.302~311にある。
- [2] J. M. Sakoda, DYSTAL manual pp.308 (Sociology Computer Laboratory Brown University, Providence, Rhode Island, 1965)
- このマニュアルを入手するには、上記のaddressあてに料金3ドルに送料をそえて申し込めば、

コピーを送ってくれる旨、P R E F A C E に述べてある（ただし現在は1970年なのでそのまゝかどうか不明）。このマニュアルは instruction manual として、演習問題の解答を含めた pp. 257 の本文と pp. 258-308 の DYSTAL システムの IBM7070 FORTRAN II のリストからなっている。FACOM 230-60 への書きかえはこのリストによった。書きかえのためには、FORTRAN II → FORTRAN IV のほかに、文字型データの処理（10進 → 2進36ビット）、浮動小数点数の表現のちがひ、入出力リストと F O R M A T の型の一致、そして広報の前号の最後に述べた関数の C A L L 文の使用を I D U M M Y にする点などに修正を加えた。FACOM 230-60 用のリストが必要な方は、九大大型計算機センターの図書資料掛に相談されたい。