[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 230−60FORTRAN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あり、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。それにもかゝわらず、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	Ｉ５
2	整数	Ｉ10
3	Ｆ型実数	Ｆ13.5
4	文字型データ(１語単位)	Ａ４
5	文字型データ(テキスト)	17Ａ４
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 FLOT(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と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、ＤＹＳＴＡＬ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、サブルチンＩＮＬＯＴでＤＹＳＴＡＬ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。

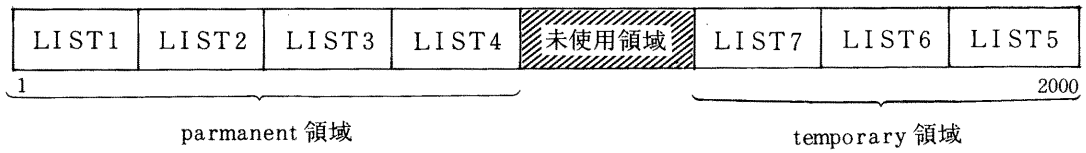

図２　　ＬＯＴ領域

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。

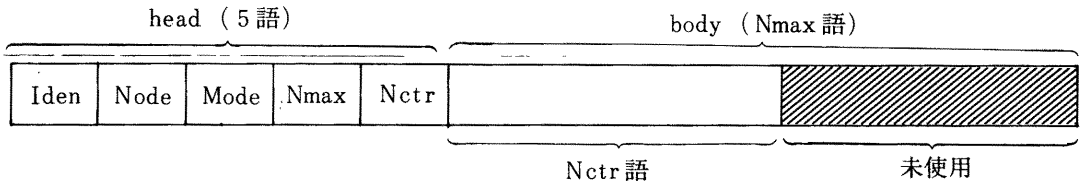

図　３　　リ　ス　ト

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は付加えたいデータ、ＬＩＳＴはリストの名である。また関数値はＩＷＤをそのまゝ返す。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 230−60ＦＯＲＴＲＡＮ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ゝ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ＩＮＴＧＥＲ）。

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う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コンパイラの特殊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（ＪＩＳｐ.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、ＪＩＳ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つの機種のＦＯＲＴＲＡＮ処理系はしらべていない）。ではＪＩＳではできもしないことをなぜ規定にするかといえば、ＪＩＳ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るべきで、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

```
IDUMMY=ISTORE (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（ＩＶＡＲ＝ＩＷＤ）という文が実行され、データの換算もなく意図どう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

```
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であることが望ましければ

```
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Iden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

```
FUNCTION LABEL (LIST, IDEN)
```

によって、ＬＩＳＴのIden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、Idenセルに▼NAME▼というラベルをつけるには、

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

この場合、このリストＬＩＳＴＮはparmanent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はサブルチンでもよいのだが、Sakoda は、関数の重畳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、

```
IDUMMY=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parmanent領域に生成する。

```
IDUMMY=LABEL (LSTALL (1, 15, LISTN), 4HLSTN)
DO 1 I=1, 15
IDUMMY=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）のようにはじのリストの場合はそのまゝ削除されるが、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。

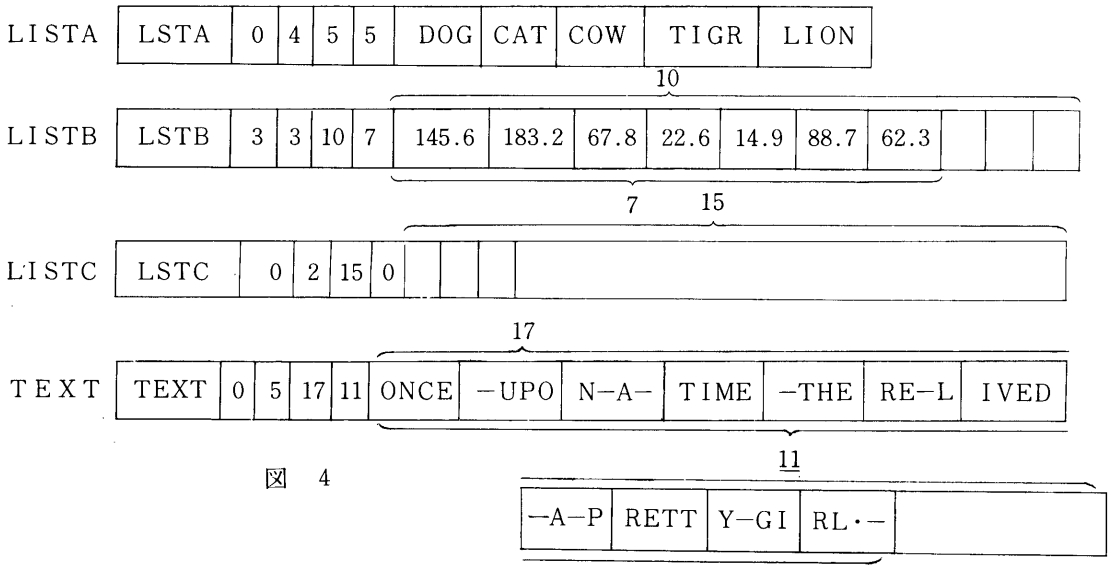

図 4

```
          Iden          Node          Mode          Nmax          Nctr
 2        8       14 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           COW           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、リストの名をエント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parmanent 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。

さてこれらの標準入力のためのＦＯＲＭＡＴは先に述べたように、図３の２番目のリストに格納されている（図６）。Mode ｉに対応するＦＯＲＭＡＴは、ＬＯＴ（ＬＦＭＴ＋４＊（ｉ－１）＋１）～ＬＯＴ（LFMT+4＊（ｉ－１）＋４）に入っているので、ＬＳＲＥＡＤが補助

図　６

```
Iden Node Mode Nmax Nctr   4×4文字
LFMT      5.   24. 24. (5(9X, I5))
                        (5(4X, I10))
                        (5(1X, F13.5))
                        (5(10X, A4))
                        (1X, 17A4)
                        (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-2)-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  MODE =     5  NMAX =    24  NCTR =    24

(5(9X,I5))     (5(4X,I10))     (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 NODE =     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）要素を取りだすには、

```
    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、まぜ合せ

なお、もともとのＤＹＳＴＡＬ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
DO20I＝1，N
IX＝LISTA＋I
SUMX＝SUMX＋FLOT（IX）
SUMXX＝SUMXX＋FLOT（IX）＊＊2
20CONTINUE
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、たゞし整数のリスト。

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、ＰＲＥＦＡＣＥに述べてある（たゞし現在は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ＩＢＭ7070ＦＯＲＴＲＡＮⅡ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→ＦＯＲＴＲＡＮⅣ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い、入出力リストとＦＯＲＭＡＴの型の一致、そして広報の前号の最後に述べた関数のＣＡＬＬ文の使用をＩDUMMY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。