JAERI-M
8715

データプールの概念と機能

1980年3月

富山　峯秀・滝川　好夫*・吉森　正大**
荻津　　実**・浅井　　清

日本原子力研究所
Japan Atomic Energy Research Institute

この報告書は，日本原子力研究所がJAERI-Mレポートとして，不定期に刊行している研究報告書です．入手，複製などのお問合わせは，日本原子力研究所技術情報部（茨城県那珂郡東海村）あて，お申しこしください．

JAERI-M reports, issued irregularly, describe the results of research works carried out in JAERI. Inquiries about the availability of reports and their reproduction should be addressed to Division of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken, Japan.

JAERI－M 8715

データプールの概念と機能

日本原子力コード委員会
総合化専門部会
富山峯秀[+]・滝川好夫[*]・吉森正大[**]
荻津　実[**]・浅井　清[+]

（1980年1月28日受理）

本報告は，この数年間原研で開発が進められてきたデータ取扱いのための汎用計算機プログラムであるデータプールの概念と機能について述べたものである。データプールの第1の目標は，データをプログラムから分離独立させてプログラムとデータのモジュラリティを高めること，プログラム間のデータの整合作業を容易にすること，などである。また，データプールはモジュラ・コード・システムの構築にも有用である。機能については，バッチ処理にかかわるものとタイムシェアリング処理にかかわるものの2つの部分にわけて説明した。データプールはこの一年間，いくつかのデータ・ライブラリ作成のため試用に供され，利用者の協力によって使用上の問題点もほぼ解消されたので，報告書第1版としてまとめ，一般の実用に供することとした。

+　東海研究所計算センタ
*　日本ソフトウェア開発㈱
**　日本ＩＢＭ㈱

JAERI-M 8715

Datapool; Its Concept and Facilities

Mineyoshi TOMIYAMA[+], Yoshio TAKIGAWA[*],
Masahiro YOSHIMORI[**], Minoru OGITSU[**],
Kiyoshi ASAI[+]

Modular Code Programming Subcommittee(JAERI),
Japan Nuclear Code Committee

(Received January 28, 1980)

In this report the authors have presented a concept and facilities of a computer program named Datapool.

Datapool is designed to handle data independently of Fortran programs which produced the data. Data in files stored by Datapool routines can be accessed easily without any knowledge of original programs. Thus the users of Datapool are able to promote modularity of their programs and data.

Datapool also comprises useful facilities as a component of modular code systems.

Keywords: Computer, Database, Datapool, Modular Coding, Data Handling

---

+ Computing Center, JAERI
* Nippon Software Kaihatsu, Ltd.
** IBM, Japan

目　　次

Contents

1. データプール概説

1.1 はじめに

本報告において筆者らはFortran プログラムと連動するデータプール・ファイルとそのプロセッサについて述べる。一般に科学技術計算のプログラム・ファイルは，計算の状況によってデータ長や書込み順序が変化する場合がほとんどである。このために作成済のデータ・ファイルを使用するときは，関連あるプログラムの調査を，その入力データまで含めて必要とすることが多い。すなわち，データ・ファイルは計算プログラムから独立ではない。一方当研究所においては大量のデータを取扱う大型プログラムが多数使用されており，研究者はこれらプログラム間のデータの整合作業に多大の時間を費やしてきた。Fortran プログラムとそのデータ・ファイルの持つこの難点を解消し，データ・ファイルのモジュラリティを高め，データの標準化と共用を容易にすることがデータプール開発の目的である。

この目的に合致するデータプール・ファイルのもつべき機能を，

樹状階層構造の命名法，データおよびデータ属性の保存，データの内容，用途などについての注釈文の保存，

とした。また，このファイルについて

データおよび属性の検索，表示，修正，消去，保管と管理などの操作が容易におこなえる，

ことをファイル処理ユーティリティの持つべき機能とした。さらに，これらの機能がすべてFortran 語の枠を出ないこと，その処理系がすべてFortran 語で書かれていることを条件とした。

1.2 データプールの機能

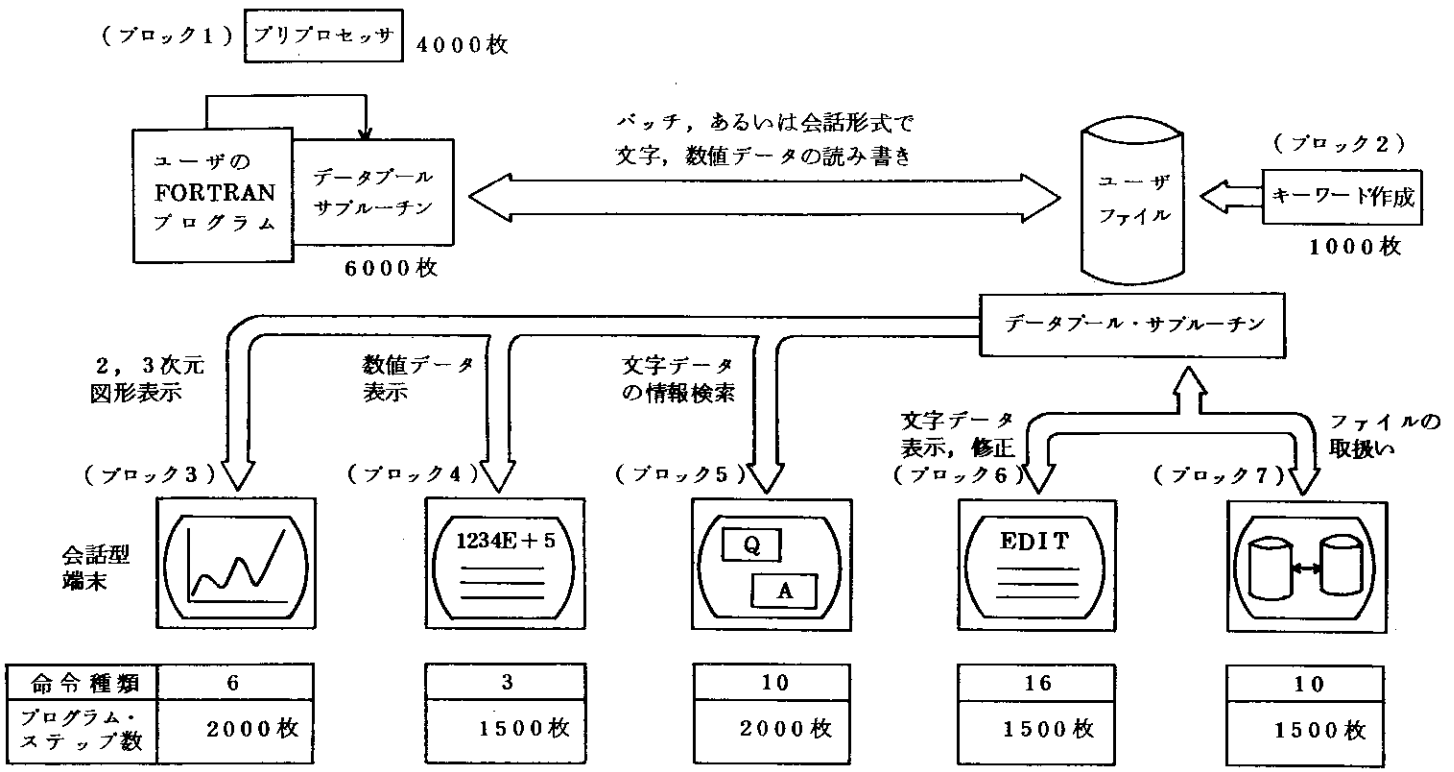
1.2.1 プリプロセッサ

筆者らはFortran プログラム中にあってデータの入出力をおこなうデータプール入出力文を定め，このデータプール用入出力文を通常のFortran 入出力文へ自動的に変換するプリプロセッサを用意した（図1.1，ブロック1）。これによってデータプールを使用する場合に，プログラム変換の手間を最小限にとどめることができる。プリプロセッサは，ソース・プログラム・カードの第1，2カラムからCJではじまる文（以後CJ文と呼ぶ）を解読し，これを通常のFortran 文に展用する。展用例を図1.2に示す。例からわかるとおり，入出力はFortran の直接アクセス文を利用しておこなう。

1.2.2 入出力アクセス法

データプール・プロセッサのデータへのアクセスはつぎのようにしておこなわれる。

(i) データプール・プロセッサは，空白（ブランク）の文字列のデータ名に出会ったとき，あるいは木構造の終端のノードに到着したときに指定されたデータ名の探索を終了する。この



命令種類	6	3	10	16	10
プログラム・ステップ数	2000枚	1500枚	2000枚	1500枚	1500枚

注

プログラムはすべてFORTRAN で書かれている。プログラム・ステップ数は実行文のみで，コメント文は含まない。

Fig. 1.1 Schematic view of Datapool System

```
      SUBROUTINE DPW(A,B,C,D,X,N)
      DOUBLE PRECISION A,B,C,D
      DIMENSION X(100)
CJ    WRITE(A,B,C,D,ERR=200) (X(I),I=1,N)
      RETURN
  200 STOP
      END
```

⇩

```
      SUBROUTINE DPW(A,B,C,D,X,N)
      DOUBLE PRECISION A,B,C,D
      DIMENSION X(100)
C=====================================================================
CJ    WRITE(A,B,C,D,ERR=200) (X(I),I=1,N)
C=====================================================================
      ILNGTH=(1*(N-1+1)+0)
      CALL GOPEN(4,2,ILNGTH,IUNIT,IRECRD,0,4,1,0,A,1,0,B,1,0,C,1,0,D,1,&
     1200)
      CALL GWRITE(IRETCD,0,12,'(X(I),I=1,N)',4,510,100,400,N,1,&200)
      WRITE(IUNIT'IRECRD,ERR=200)(X(I),I=1,N)
      CALL GCLOSE(IRETCD)
C=====================================================================
      RETURN
  200 STOP
      END
```

Fig. 1.2 Expansion of CJ Statement

機能を利用すると，データプール・ファイルのすべてのデータ名をひとつの入出力文でアクセスすることができる。

つぎの出力文

CJ WRITE (A.B.C. …. X) (DATA(I), I = 1, N)

において，データ名の変数C, …, X のすべてが空白（ブランク）文字なら上記の出力文は

CJ WRITE (A. B) (DATA(I), I = 1, N)

と同じである。

(ii) 利用者はデータプール・ファイルの木構造を水平，あるいは垂直に探索することができる。

つぎの例

```
CJ POINT ( A.B.C, MAXD, ND, DNAME, NRC )
CJ NEXTV
CJ READ ( *, ERR=123 ) U, V
CJ NEXTH
CJ READ ( *, ERR=123 ) U, V
```

においてＰＯＩＮＴ文は探索開始点となるデータ名を示す。この場合はA.B.Cである。どのデータ名が探索の対象となっているかはノード位置設定ポインタが示している。ＮＥＸＴＶはデータ名A.B.Cのひとつ下の階層で先頭のデータ名に位置設定ポインタを移し，そこのデータ名が配列変数ＤＮＡＭＥに格納される。ＲＥＡＤ文の＊印指定によって，このデータ名のデータを読むことができる。ＮＥＸＴＨ文は位置設定ポインタを水平右方向に移し，そこのデータ名を配列ＤＮＡＭＥに格納する。

1.2.3 入出力並びの保存

データプール・プロセッサは，データプール・ファイルの入出力時にデータの入出力並びとその並びに現われた変数の型，値などの属性を記録する（図1.3）。

通常は自動的に記録されるが，使用されるダイレクトリ数を減らしたいときは省略することができる。その選択は入出力文中に指定された優先順位Ｐの値による。データプール中のデータを読み書きするとき，前回の入出力並びと今回使用する入出力並びとの整合性のチェックはおこなわれない。データが書込まれたときは，そのデータの出力並びに更新される。

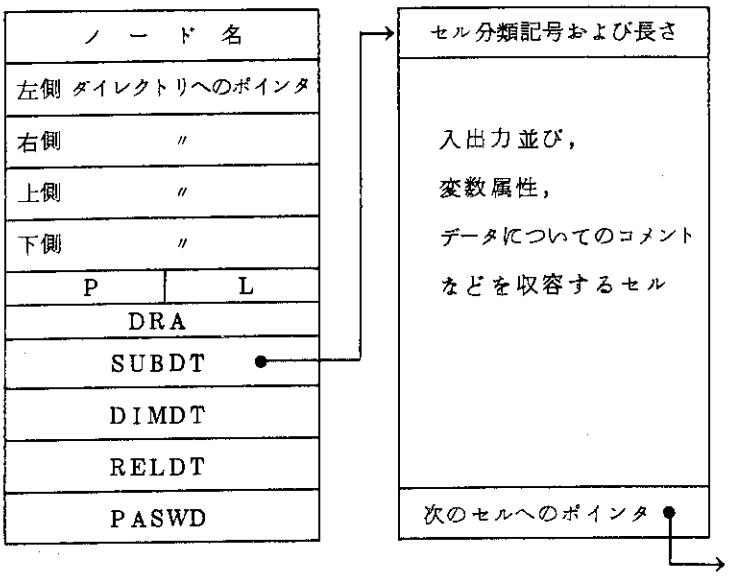


Fig. 1.3 Cells for Data Name and Data Attributes

1.2.4 注　釈

データプールの利用者は，バッチあるいはタイムシェアリング処理でデータに注釈を与えることができる。注釈は図1.3に示すようにデータのダイレクトリに結合されている。注釈は20語（80バイト）単位で伸縮自在となっており，タイムシェアリング端末で編集することができる。注釈の文字列の形式を1.2.5で述べる情報検索の様式に合わせておけば，データについての情報検索システムを容易に作ることができる。

1.2.5 情報検索

データプールは保存するデータについての簡易な検索機能をもつ（図1.1，ブロック5）。情報検索機能は，検索用キーワード表の作成をバッチ処理でおこなうプログラムと，会話形式で情報検索をおこなうプログラムとから成る。検索は，キーワードをオペランド（文字列，またはキーワード番号）とする論理式を端末から入力することによっておこなう。使用例を図1.4に示す。図1.4の論理式オペランドのキーワード番号は，前述の検索用キーワード表作成プログラムによって作成される（図1.1，ブロック2）。この検索システムは，ごく小規模なキーワードの集合を検索の対象としており，取扱えるキーワード総数は現在のところ2000程度としている。データの大きさ，データ名（ノード名）の数は，これに関係なく大きくとることができる。

```
KEY(P)
>PADE*APPROXIMANT +3
 ABUE      ACRS      ACRT      ACWF      ACWG
LIST(P)
>ACRS
**V:09,1975,46-50                C:ACRS,ICL1430,343,EBCDIC
T:A SUBROUTINE AND PROCEDURE FOR THE RAPID CALCULATION OF SIMPLE OFF-
  DIAGONAL RATIONAL APPROXIMANTS
A:P.R.G.MORRIS,D.E.ROBERTS
K:GENERAL,RATIONAL,APPROXIMANT,CHISHOLM,SIMPLE OFF-DIAGONAL,PRONG,
  PERTURBATION SERIES,PADE
AC:PROGRAM ACRS IS A FORTRAN VERSION OF SODS
LIST(P)
>#END
```

Fig. 1.4 Example of Information Retrieval

1.2.6 データの図形表示

データプール・ファイルに保存されているデータを図形表示端末に出力することができる（図1.1，ブロック3）。Fortran の出力並び中の任意の変数について2次元，3次元の表示をおこなうことができる。配列変数は表示の際に初期値，終値，増分値を指定することができ，またグラフに複数本のタテ軸の追加，軸ラベルの設定，タイトルの記入などもできる。2次元図形の場合には，X，Y軸はそれぞれ2350点まで一度に表示でき，3次元図形の場合にはX，Y軸は50点，Z軸は2500点まで一度に表示できる。端末装置はいずれもテクトロニクスT4000シリーズを対象にしている。使用例を図1.5に示す。

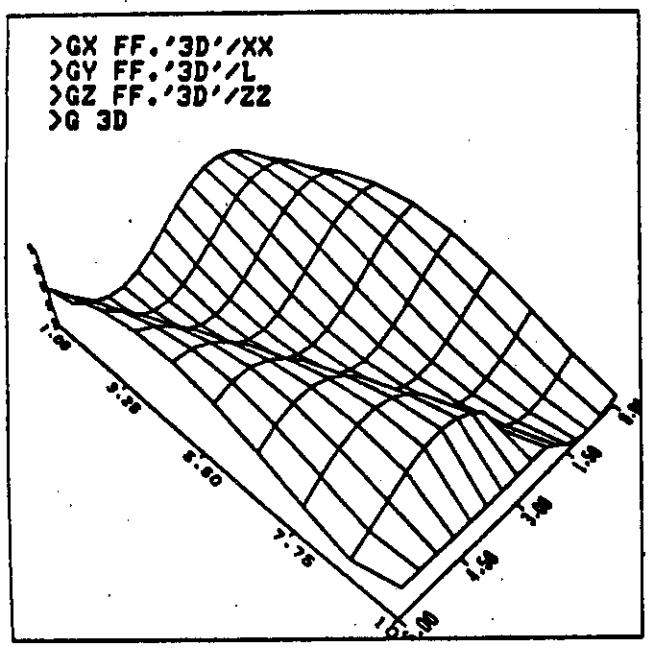


Fig. 1.5 Example of Graphical Display

1.2.7　データの数値表示

データプールに保存されているデータを端末に数値表示することができる（図1.1，ブロック4）。図形表示と同じく，Fortranの出力並び中の任意の変数を表示することができる。これらの表示はデータがファイルに書込まれるときに同時に記録された入出力並びを解読することによっておこなわれる。表示の対象となる変数の型は，現在のところ整数，実数，倍精度実数，虚数に限られる。出力の対象となったデータ名（ノード名）の入出力並びも同時に出力する。配列変数は，図形表示の場合と同様に，初期値，終値，増分を指定することができる。表示例を図1.6に示す。

1.2.8　注釈の入力，表示，修正

データプールでは，データに注釈をつけて保存することができる（図1.1，ブロック6）。注釈は，データプールを使用しているプログラムの実行中にデータプール・ファイルに書込むことができる。また，会話形式でも注釈の入力，表示，修正などが可能である。例を図1.7に示す。

1.2.9　ファイルの取扱いとコマンド・プロシジャ

データについて，名称の変更，消去，移動，複写，比較などの操作ができる。また，データプール・ファイルについての定型的操作を容易にするために，簡易なコマンド・プロシジャの機能を有する（図1.1，ブロック7）。コマンド一覧表を表1.1に，使用例を図1.7に示す。

1.2.10　データプール・ファイルの保守と管理

データプール・プロセッサは次のような保守と管理の機能をもっている。

(i)　RECOVERコマンド

データプールにデータを書込み中のプログラムが不測の障害により実行打切りとなったときは，データプール・ファイルのクローズが完了せず，データプールの管理テーブルが乱れデータの追加不可能となる場合がある。RECOVER機能はデータプール・ファイル内の管理テーブルを補修し，当該データプール・ファイルを再使用可能とする。

(ii)　SORTコマンド

データプールの樹状階層構造はデータ名の保存順に構成される。SORTはデータ名を単位として昇ベキ，あるいは降ベキの順に並べかえる。

(iii)　CONDENSEコマンド

データ名の消去などにより使用不可となったデータ領域を集め1つの使用可能領域とする。

```
>XYA= 'XYALL'
>LISTD FF.XYA/XX
*** I/O LIST ***
Y2,N,(YY(I),I=1,N),I1,I2,(XX(J),J=I1,I2)
*** CONTROL VARIABLES ***
*** SYMBOLS ***
XX
*** DATA ***
 0.0              0.4000000E-01  0.8000000E-01
 0.3200000E+00    0.3600000E+00  0.4000000E+00
 0.6400000E+00    0.6800000E+00  0.7200000E+00
 0.9600000E+00    0.1000000E+01  0.1040000E+01
```

Fig. 1.6 Example of Numesical Display

```
00010        PROCEDURE TESTPRO
00020        ATTACH F01,J9131.TESTPOOL,LRECL=80/U,MAXRCD=500
00030        ON ERROR STOP 'ERROR STOP'
00040        SYNONYM FILE='J9131'.'TESTPOOL'.'COMMENT'.#NN
00050        NN = 0
00060    100 NN = NN + 1
00070        CAREA 80,A(1)
00080        CW FILE
00090        ON (NN.LT.9) GOTO 100
00100        MANUAL  'RETURN TO MANUAL MODE'
00110        END

#.DPSRUN PROC=TESTPRO

>RUN TESTPRO
.START ***TESTPRO *** PROGRAM
.START TEXT MODE
#....*....1....*....2....*....3....*....4....*....5....*....6.
>TESTCOMMENT CREATED. DIMENSIONAL CELL #1
>$END
.END TEXT MODE
```

Fig. 1.7 Example of Comment and Cataloged Procedure

Table 1.1 Commands for TSS

コマンド	省略形	機能
ATTACH	A	指定されたファイルを割当てる
ALLOCATE	ALLO	新しくファイルを作る
CATLIST	CATL	ファイル，あるいはデータの属性を表示する
PROCEDURE	PROC	プロシジャ宣言をおこなう
RUN	RUN	指定されたプロシジャを実行する
END	END	プロシジャ宣言と実行の終了を示す
*	*	注釈文を示す
GO TO	GO TO	指定されたラベルの文へジャンプする
STOP	STOP	プロシジャの実行を終了させる
MANUAL	MANUAL	処理系を手動状態にする
ON	ON	指定された条件を検出する
SYN	SYN	ファイル名，あるいはデータ名の同義語宣言をおこなう
代入文		数値，データ名などの代入操作をおこなう
COMMAREA	CAREA	データについてのコメント編集用領域の宣言
テキスト		データについてのコメント用文字列
¥END	¥END	コメント用テキスト入力の終了を示す
DISPLAY	DISP	CAREAの内容を表示する
COMMR	CR	指定されたデータ名のコメントを読む
COMMW	CW	指定されたデータ名へコメントを書く
COMMWA	CWA	指定されたデータ名へコメントを書き加える
COMMD	CD	指定されたデータ名のコメントを消去する
LIST	L	指定されたデータ名の属性を表示する
COMPARE	COM	指定された2個のデータ名の内容を比較する
MOVE	MOVE	指定されたデータ名の内容を他のデータ名へ移す
COPY	COPY	指定されたデータ名の内容を複写する
DELETE	DELETE	指定されたデータ名とその内容を消去する
RENAME	R	指定されたデータ名を改名する
LISTD	LD	指定されたデータ名と変数の内容を表示する
GRAPH		指定されたデータ名と変数の内容を図形表示する

1.3 ファイルとプロセッサの内部構造

1.3.1 プリプロセッサの機能

プリプロセッサは，Fortran ソース・カードの第1，2欄に文字C，Jを穿孔してあるCJ文で記述されたデータプール・ファイルへの入出力文を Fortran 文に展開する。展開された入出力文などはデータプール・アクセスのためのサブルーチン呼出し文のパラメータ数が，データ名の長さによって違ってくる。そのため該当するサブルーチンは，Fortran 語ではなくシステム記述言語GPL[1)]で書かれている。このようなサブルーチンは10個存在し，個々の大きさはそれぞれ約20ステップである。実際に必要なサブルーチンは，このGPLで書かれたサブルーチンから呼出される。

プリプロセッサもGPLで記述されている。このプリプロセッサは，もともとは会話形 Fortran の構文解析，およびBackus形式テキストの生成用に作られたプログラムから，DO文，IF文，代入文，書式文などの解析機能を取去り，残った部分にCJ文解釈部とその展開部をつけ加えたものである（図1.1，ブロック1）。GPLで約4500ステップであるが，現在 Fortran 語に書換え中である。

1.3.2 データプール・ファイル

データプール・ファイルは，本質的には Fortran のランダム・アクセス・ファイルである。したがって利用者は，1つのプログラム中で任意本数のデータプール・ファイルを使用することができる。各データプール・ファイルは，そのファイルに収められるべきデータの入出力処理にもっとも都合のよいブロック長（利用者指定）で仕切られ，その先頭部分にファイル制御情報が収められている。ファイル制御情報のおもなものは，ファイルの属性を記述するファイル制御表，およびデータの属性を記述するダイレクトリ群の2つである。ファイル制御表は1本のデータプール・ファイルについて1個作られ，50語（200バイト）の大きさを持ち，そのおもな内容は表1.2のとおりである。

Table 1.2 DPCTL — Datapool File Control Table

1	'19'	21		41	MAX CASE NO.
2	'31'	22		42	CURRENT CASE NO.
3	DATA SET NAME	23	RECORD SIZE	43	POINTER FOR CASE NO. ROOT
4		24	MAX RECORDS	44	
5		25	USED RECORDS	45	
6		26	POINTER FOR UNUSED RECORDS	46	
7	CREATOR'S NAME	27		47	
8		28		48	BYTE OR WORD
9	CREATION DATE	29	MAX DIRECTORIES	49	
10		30	USED DIRECTORIES	50	
11	EXPIRATION DATE	31	POINTER FOR DIRECTORY ROOT		
12		32			
13	PASS WORD	33	POINTER FOR UNUSED DIRECTORIES		
14		34			
15	COMPUTER NAME	35	POINTER FOR SHARABLE USERS		
16	DD SET DEVICE NAME	36			
17		37	HASH TABLE SIZE		
18		38	PTR FOR HASH TABLE		
19		39	HASH CLUSTER SIZE		
20		40	PTR FOR HASH CLUSTER		

1.3.3　ダイレクトリ

ひとつのダイレクトリは２０語（８０バイト）から成るセルで，データ名，データ属性，注釈の保存などに使用される。利用者はデータプール・ファイルの創成時に，これらの情報を保存するに十分なダイレクトリ個数を指定しなければならない。

ダイレクトリの内容を示す図 1.3 において

(1)　Ｐ：データの優先順位（利用者指定）を示し，その値は，データの破壊防止，出力抑制などに利用される。

(2)　Ｌ：書込まれたデータ長（ブロック数）を示す。一度書込まれたデータの上に新しいデータを重ねて書く場合には，データプールのプロセッサは新しいデータの長さとＬとを比較する。Ｌが大きければ古いデータの上に新しいデータが書かれ，Ｌが小さければデータプール・プロセッサは連続する新しいブロックを用意し，新しいデータをそこに書込む。新しいブロックが用意された場合は，古いデータ・ブロックは再使用可能なデータ領域としてファイル制御表に登録される。いずれの場合もＬは更新される。

(3)　ＤＲＡ：磁気ディスク上のデータの先頭番地（ブロック番号）。

(4)　ＳＵＢＤＴ：入出力並び，注釈などを保存するダイレクトリへのポインタ。

(5)　ＤＩＭＤＴ：データ名に１次元の添字を与え，この添字によってデータを識別することができる。ＤＩＭＤＴは，この添字名を保存するダイレクトリへのポインタである。この機能を利用すると，ひとつのデータプール・ファイルに対し，ランダム・アクセス，シーケンシャル・アクセスの両方のアクセス法が可能となる。添字つきデータ名使用の１例をあげると次のようになる。

```
      DO  10  I = J, K, M
CJ    WRITE  ( A.B. #I )   I/O  list
   10 CONTINUE
```

ここで，A.BとA.B.#Iとは異ったデータを指し，またＩの内容はそれが正整数であれば何でもよい。

(6)　ＲＥＬＤＴ：このデータ（ノード）名と関連あるダイレクトリへのポインタである。データプール・ファイルは樹状階層構造を採用しているので，異った分岐ノード間相互の関連づけ機能がない。この欠点を補うために用意されたポインタである。

(7)　ＰＡＳＷＤ：このデータの読み書きに関するパスワード。現在は未使用。

1.3.4　データ名へのアクセス

データへの速いアクセスを第１義に考えれば，ダイレクトリは主記憶上に常駐していることがのぞましい。しかし実用上はダイレクトリが数万個になることもある。この問題を解決するために筆者らのデータプール・プロセッサでは図 1.8 のようなＬＲＵ（ Least Recently Used ）テーブル，ＤＴ（ダイレクトリ）テーブル，ＤＴバッファを設けている。ＬＲＵテーブルとＤＴテーブルはそれぞれ３０個のセルから成り，複数のデータプール・ファイルに共通して使用される。ＤＴテーブル上のダイレクトリはディスク上のダイレクトリのコピーに加えて，さらに５個のポインタが追加される（図 1.9 ）。このポインタはＤＴセルの番地を指して

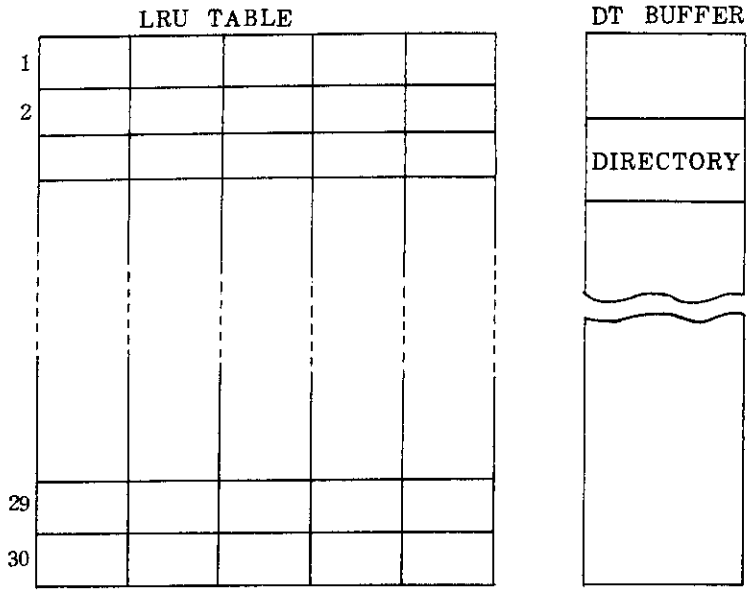


Fig. 1.8 LRU Table and Buffer

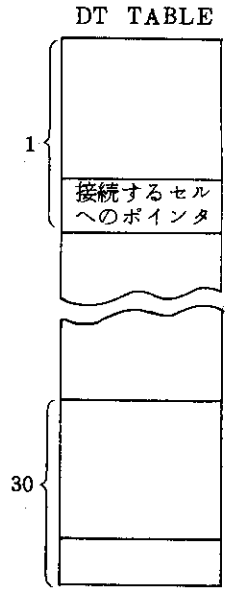


Fig. 1.9 Directory Table

おり，これによって現在アクセス中のノード群の局所的な木構造を保存することができる。DTバッファは各データプール・ファイルに1個用意され，ダイレクトリを含む1ブロックが読込まれる。LRUテーブルの第i番目のセルにDTテーブルの第i番目のセルが対応し，これらのセルはLRU方式で管理される。管理の方法は，

(i) 一番最近参照されたノードに最大の参照頻度値30を与える。LRUテーブルの他のノードについての参照カウンタの値を1だけ減ずる。値がゼロとなっているカウンタはそのままにしておく。

(ii) LRUテーブルにない新しいノードが参照されたときは，参照頻度カウンタの値が最小のノードを追出す。参照頻度カウンタの値が同じノードなら階層の一番深いノードを追出す。

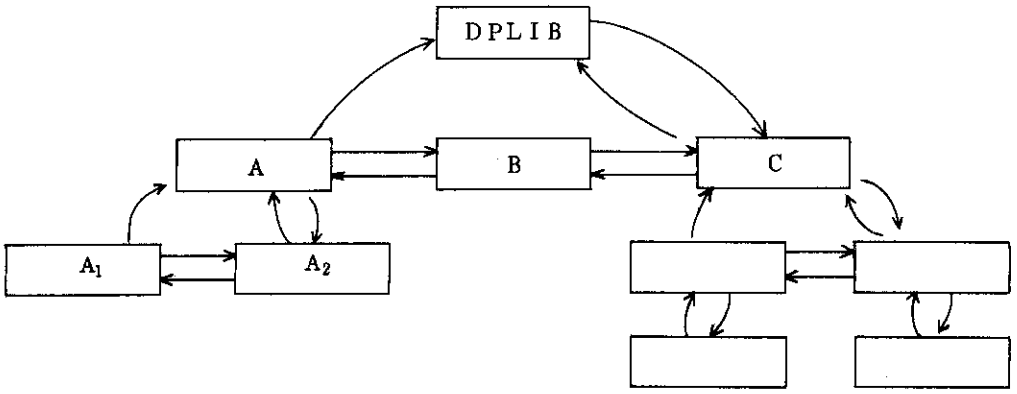


Fig. 1.10 Example of Datapool File Structure

このようなアルゴリズムでは図1.10においてノードA，CはLRUテーブルに残りやすく，A_1，A_2…，Bなどは消去されやすい。データプール利用者の希望するデータ・アクセス方法について検討した結果，このようなノード管理の方式を採用することになった。科学技術計算の分野ではこの種のデータ・アクセス法が多いのではないかと筆者らは考えている。

1.3.5 変数へのアクセス

前節のアクセス法はノード，すなわちデータ名を単位とするものである。ここではデータ名と変数名が同時に指定された場合のアクセス法について簡単な例で説明しよう。

データ名がD，入出力並びが（A(I)，B（I，J）I＝1，M，J＝1，N）で書込まれているデータ中の変数Bの1部を1つおきに会話形端末で数値表示するときのコマンドは，

LISTD　D／B，1，M，2

である。ここで，1は出発値，Mは終値，2は増分を表わす。表示ルーチンはデータ（A(I)，B（I，J），I＝1，M，J＝1，N）を作業領域に読込みながら，データ（B（I，1），I＝1，M）を抽出して出力する。出力可能なデータの型は，整数，単精度，倍精度，4倍精度の実数と虚数であるが，図形表示の場合にはこれらのデータはすべて自動的に単精度に変換して利用される。入出力並びの解析方法は，Fortranコンパイラと類似の方法に依っているが，インタプリティブな実行が要求されるため，コンパイラのそれよりはやや複雑である。紙数の都合で内容の説明は省略する。

1.3.6 検索用論理式の計算

(1) 検索式のチェック

会話形端末から入力された文字列をチェックし，入力演算子のスタックSOP，キーワードの表SKWを作る（図1.11）。これらSOP，SKWの内容を，表1.3にもとづいて検査し，演算子やキーワードの隣接状態が正しいかどうかをみる。これとは別に入力された左右のカッコのバランス，順序についてもしらべる。このときSKWの各キーワードが，登録されている正規のものかどうかについても同時にしらべる。表1.3で＊は論理積を，＋は論理和を，＜は入力文の始端を，＞は終端を示す。×は間違った組合せを，Oは正しい組合せを示す。

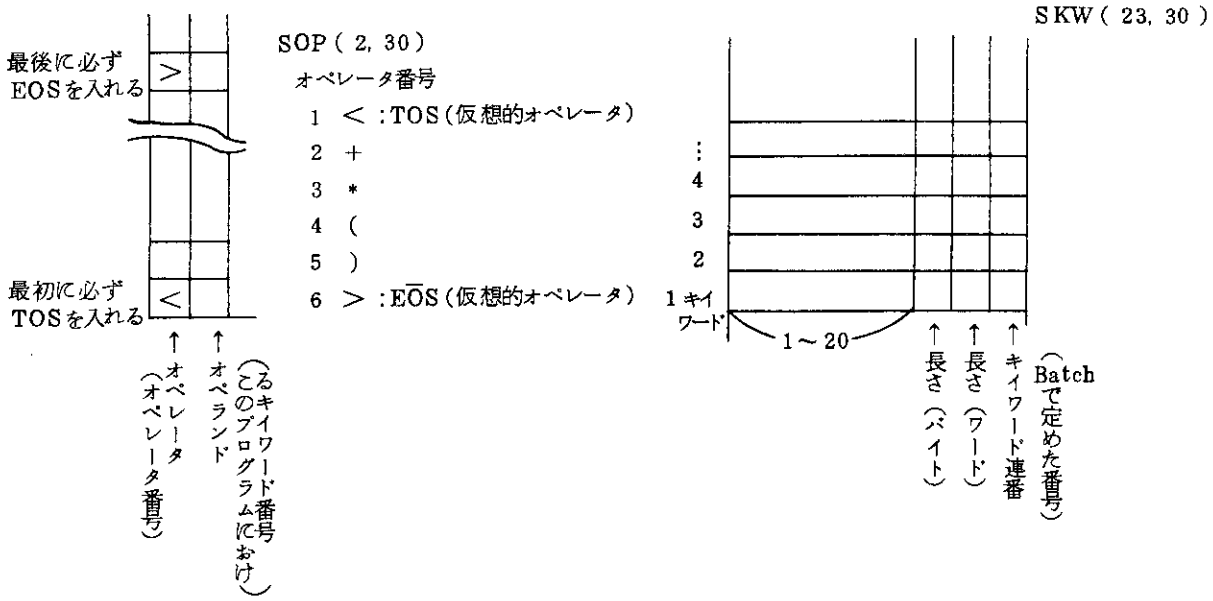


Fig. 1.11 SOP Stack and SKW Table

Table 1.3 Syntax Check Table for Logical Expression

TBLS (7, 7)

左側 ＼ 右側		1 <	2 +	3 *	4 (	5)	6 >	7 キイワード
1	<	×	×	×	○	×	×	○
2	+	×	×	×	○	×	×	○
3	*	×	×	×	○	×	×	○
4	(	×	×	×	○	×	×	○
5	)	×	○	○	×	○	○	×
6	>	×	×	×	×	×	×	×
7	キイワード	×	○	○	×	○	○	×

(2) 検索用論理式の実行

前項(1)のチェックの結果エラーが発見されなかったときは演算を実行する。演算は表 1.4 の指定に従っておこなう。表 1.4 において×は起り得ない組合せを，Sは左側の演算子に関する操作のスキップを，Eは演算子の実行を，Oは式についての演算の終了を示す。

Table 1.4 Execution Instruction Table for Logical Operation

TBLP(6, 6)

左側 ＼ 右側		1 <	2 +	3 *	4 (	5)	6 >
1	<	×	S	S	S	×	O
2	+	×	E	S	S	E	E
3	*	×	E	E	S	E	E
4	(	×	S	S	S	O	×
5	)	×	×	×	×	×	×
6	>	×	×	×	×	×	×

2. FORTRAN拡張文

この章ではデータを樹状階層構造で命名する方法と，データプールへの入出力の手段として使用されるFORTRAN拡張文について述べる。FORTRAN拡張文は文の第1桁からCJという文字列で始まり他の文と区別されているのでCJ文と呼ばれる。CJ文はデータプールへアクセスするFORTRANプログラム中に書かれる。CJ文を含むプログラムはそのままFORTRANコンパイラで翻訳すると，CJ文が注釈行として処理されてしまうので，翻訳前にプリプロセッサによりCJ文をデータプールへアクセスするためのFORTRAN文に変換しておく必要がある。CJ文は大別するとプリプロセッサにより数行のFORTRAN文に展開される「実行文」と展開のための情報を与える「宣言文」に分けられる。さらに実行文はデータプールの定義を行う「データプール定義文」，データや注釈文の入出力を行う「入出力文」，ノード名の検索を行うための「ノード位置設定文」，データプール・ファイルの保守管理を行う「保守管理文」に分けられる。表2.1にFORTRAN拡張文の一覧表を示す。

Table 2.1 Fortran Extended Statements for Datapool

区分	FORTRAN拡張文名	パラメータ	機能概要	頁
定義文	FD	Fnn＝データプール名, LRECR＝レコードサイズ／{L/E/U}, MAXRCD＝最大レコード数 〔, MODE＝{OLD/NEW}, OWNER＝所有者名, DIRECT＝ディレクトリの個数 〔, MACHINE＝計算機名／{WORD/BYTE}〔, DEVICE＝装置名／レコードギャップ／1トラック当りのバイト数〕〕	使用するデータプールの属性を定義する。	
入出力文	WRITE	({データ名/*/#}〔, P＝優先権〕〔, EXT＝展開形式〕〔, FMT＝書式名〕 〔, ERR＝文番号〕) 出力並び	データプールにデータを格納する。	
	READ	({データ名/*/#}〔, ERR＝文番号〕) 入力並び	データプールからデータを読む。	
	COMMW	({データ名/*/#}〔, P＝優先権〕〔, ERR＝文番号〕) {文字列/n, 変数名}	注釈文を格納する。	
	COMMR	({データ名/*/#}〔, ERR＝文番号〕) n, 変数名	格納されている注釈文を読む。	
	SEARCH	(データ名, データ長〔, ERR＝文番号〕)	指定されたデータ名をさがす。	
ノード位置設定文	POINT	(データ名, D_{max}, D, v, rc)	ノード名探索の開始点を指定する。	
	NEXTH		ノード位置設定ポインタを水平方向右側に移動する。	
	NEXTV		ノード位置設定ポインタを下方向左端に移動する。	
	PRIORH		ノード位置設定ポインタを水平方向左側に移動する。	
	PRIORV		ノード位置設定ポインタを上のノードに移動する。	

区分	FORTRAN拡張文名	パラメータ	機能概要	頁
ノード位置設定文	CONNECT	(データ名1, データ名2, RC)	二つのノードを結びつける。	
	DISCON	(データ名1, データ名2, RC)	CONNECT文で作られた結びびつきを解く。	
	RNEXT		ノード位置設定ポインタを現在の位置からCONNECT文で結びつけられているノードに移す。	
	FIND		ノード位置設定ポインタが示しているノードのノード名を知る。	
	DNEXT	(変数名)	一次元インデックス位置設定ポインタを現在の位置から次の位置へ移す。	
保守管理文	CONDENSE	(データプール名, WF=nn)	データプール内の各所に散在する使用不可領域を集め，使用可能領域とする。	
	SORT	(ノード名[, LEVEL={i / ALL}][, WF=nn][, SR=m])	指定されたレベルのノード名を並べかえる。	
	RECOVER	(データプール名[, TYPE={W / {S, WF=nn} / {C, WF=nn}}])	使用不可となったデータプールを再使用可能にする。	
宣言文	OPTION	[BUFFER=作業用領域サイズ/{L / E / U},][LRUSIZE={30 / LRUテーブルサイズ},] [MAXUDP=最大使用データプール数,][QMAIN={CREATE / NOCREATE}]	データプール・アクセス・ルーチンの動作環境を設定する。	
	SYN	v1=データ名1[, v2=データ名2, ……]	同義語を定義する。	
	ALIAS	v1=予約語1[, v2=予約語2, ……]	予約語を変更する。	

2.1 FORTRAN拡張文の使用法

2.1.1 樹状階層構造とデータ名

図2.1に簡単な樹状階層構造の例を示した。図において丸印をノードと呼びその名前をノード名と呼ぶ。Aから始まりノード名をピリオドで区切って並べたものをデータ名と呼ぶ。例えば

A・B

A・B・X1

A・B・X1・Y1

A・B・X2・Y4

などがデータ名である。データプールへのデータの入出力はこのデータ名を指定して行う。

```
CJ  WRITE(A・B・X1・Y1)……                    ……①
CJ  WRITE(A・B・X1・Y2)……
CJ  WRITE(A・B・X2・Y3)……
CJ  WRITE(A・B・X2・Y4)……
CJ  WRITE(A・B・X3・Y5)……
CJ  WRITE(A・B・X3・Y6)……
```

を実行すると図2.1の樹状階層構造ができる。ここで

```
CJ  READ(A・B・X1・Y1)……
```

を実行させると①で格納したデータが読み込まれる。データは指定したデータ名の最後のノードに属して格納されている。

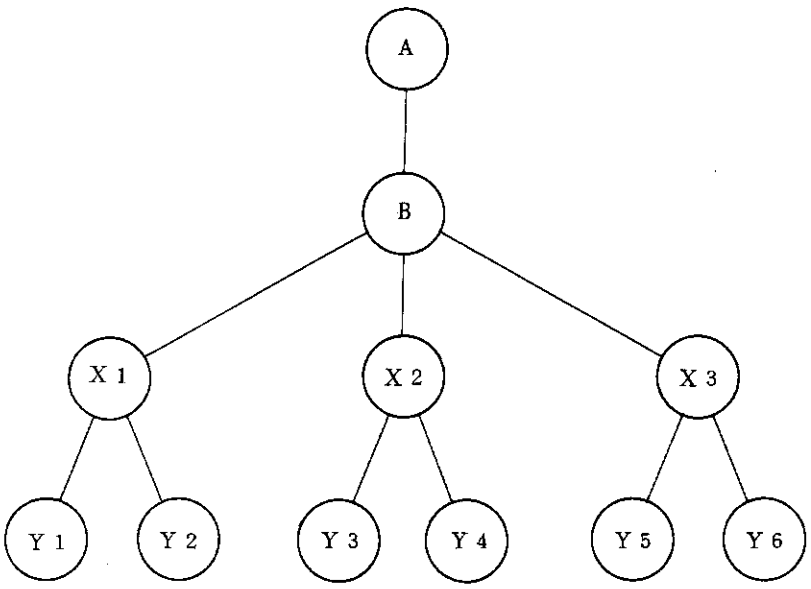


Fig. 2.1 Example of Tree Structured File

2.1.2　添字付データ名と1次元インデックス

1つのデータ名に対して数ケースのデータ（ディメンジョナル・データ）が存在する場合などはデータ名に添字を付けてデータを格納できると便利である。データ名の記述において添字は

データ名・#n

の形式で表記する。nは正の整数か変数名である。#nを一字元インデックスと呼び，格納されたデータを1次元インデックス付きデータと呼ぶ。1次元インデックスは特殊な結びつきのノードを形成する。図2.2に1次元インデックス付の樹状階層構造の例を示す。例えば

```
      DO  10  N=1, 2
  CJ  WRITE(A・B・X1・Y2・#N)……
  CJ  WRITE(A・B・X2・Y4・#N)……
   10 CONTINLE
```

を実行すると図2.2のような1次元インデックス付の樹状階層構造ができる。1次元インデックス付のデータはインデックスの指定により，順次入出力することもランダムに入出力することもできる。

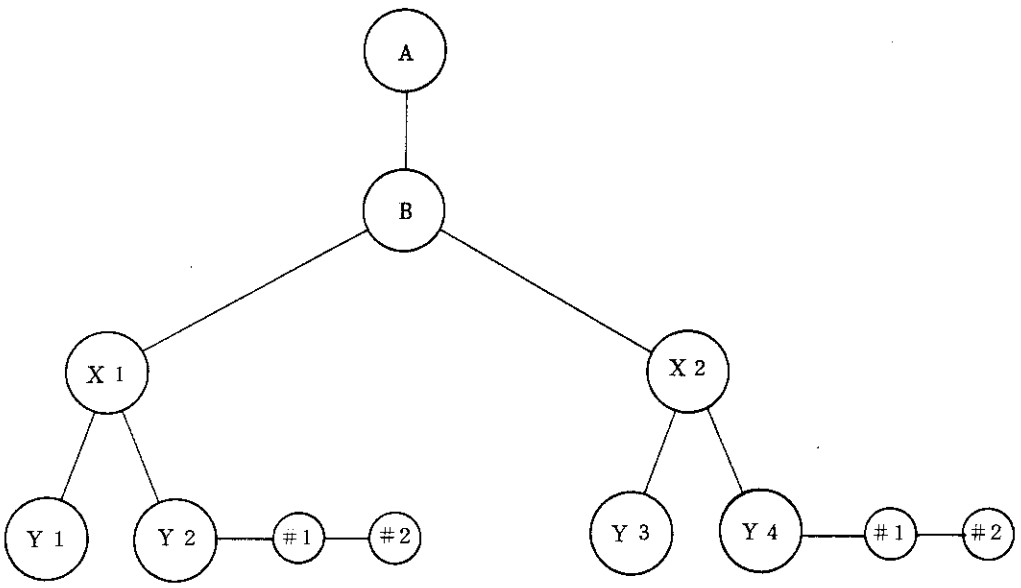


Fig. 2.2　Tree Structured File with One Dimensional Index

2.1.3　データ名記述上の注意

データ名は次の規約に従って書かなければならない。

データ名：＝ノード名1．ノード名2．……ノード名n

または

：＝ノード名1．ノード名2．……ノード名n．#一次元インデックス

ノード名：＝▼8文字以内の任意の文字列▼

または

：＝添字の付かない変数名

変数名：＝2語長（DOUBLE PRECISION）で定義されているもの

または

：＝SYN文で定義された同義語

または

：＝2語以上の連続領域名

一次元インデックス：＝#記号と正の整数

または

：＝#記号と整数型変数名

データ名のうち第1と第2のノード名を一緒にしてデータプール名またはファイル名と呼ぶ。またデータ名を構成するノードの数をノードの深さという。一次元インデックスはデータ名の添字であるからノードの深さには数えない。

樹状階層構造はWRITE文等で指定したデータ名が登録されていく順（上から下，左から右）に構成されていく。例えば図2.1において

CJ WRITE(A・B・X2・Y1)……

を実行するとノードY1はY3の左側でなくY4の右側に登録される。ただし一次元インデックスは昇べきの順に並べかえ登録される。

樹状階層構造の末端のノードをリーフ（leaf）ノードと呼ぶことがある。

実行文中のデータ名やデータプール名として指定されるノード名の並びは変数名または直接文字列をピリオドで区切って列記する。文字列で指定する場合はノード名を引用符（▼）で囲んで指定する。また変数名で指定した場合はその実行文が実行される前に変数名に対してノード名が指定されていなければならない。

実行文中に指定されたデータ名において，データ名を構成するノード名のうち，途中のノード名に空白のものがあるとデータプール・プロセッサは以後のノードを無視し，空白の直前までがデータ名として指定されたものとみなす。

2.1.4 FORTRAN言語のレベル

プリプロセッサはCJ文を含むプログラムの各文の解釈を行う。FORTRAN文法は計算システムごとに方言を持っており，全てが同一ではない。プリプロセッサは当初すべての計算機システムで利用できると考えられるJIS7000レベルのFORTRAN文法を解釈するように作成した。しかしこれは現実の利用にそぐわない面があり，現在は宣言文と入出力文以外のFORTRAN文の文法解釈は行っていない。

ただしFORTRAN拡張文のWRITE文，READ文において，入出力並びの添字式およびDO型並びの初期値，終値，増分値の式は，整数および添字の付かない変数名の四則演算式を解釈できるように拡張してある。

2.1.5　FORTRAN 拡張文の書き方

FORTRAN 拡張文（ＣＪ文）は次の形式で書かれなければならない。

① 第１桁からＣＪと続けて書く。

② 第７桁から本文を書く。

③ 継続行は第６桁に空白以外の文字を書く。

④ ＣＪ文に文番号をつけてはならない。

またパラメータについては各文の文法に従って書く。書き方は

① 大括弧で囲まれたものは省略してもよい。

② 中括弧で囲まれたものは，その中から１つを選択し記述する。

③ カナ，漢字，英小文字，イタリック体の英文字で書かれているものはそれに相当するものを文字列または変数名で指定する。

④ 特殊記号（▼（　），＊＃＝）や英大文字で書かれているものはそのまま記述しなければならない。

⑤ 下線の引かれているものはパラメータ省略時の標準値である。

2.2　ＦＯＲＴＲＡＮ拡張文文法

2.2.1　データプール定義文

2.2.1.1　ＦＤ文

1〕構　文

```
CJ    FD Fnn=データプール名,〔LRECL=〕レコードサイズ/{U/E/L}{〔MAXRCD=〕最大レコード数 / TRK=トラック数}
      [,〔MODE=〕{OLD/NEW},〔OWNER=〕所有者名,〔DIRECT=〕
      ディレクトリの個数}[,〔MACHINE=〕{▼F230−75▼/▼WORD▼ / 計算機名/タイプ2}
      ,〔DEVICE=〕{▼F479B2▼/135/13030 / 装置名/レコードギャップ/1トラック当りのバイト数}]]
```

2〕機　能

新たに創成または参照するデータプールの属性を定義する。

3〕パラメータの説明

ＦＤ文のパラメータはほとんどすべてが変数名または直接数値，文字列で指定できる。パラメータを書く順序は任意である。またFnnを除き等号とその左辺を省略できる。ただし

左辺を省略する場合は構文に従った順序で指定しなければならない。

Fnn＝データプール名

Fnn：ファイル定義名を変数名またはFに続く2桁の整数で指定する。プリプロセッサはFD文を解釈するとき，この位置に書かれているF01からF99までの文字列を変数名とはみなさない。変数名を指定した場合はプログラムの実行時にFD文が実行される前に変数名に対してFに続く2桁の整数の文字列が割り当てられていなければならない。

データプール名：樹状階層構造の第一ノードと第二ノードのノード名をピリオド(.)で区切って書く。第一ノード名を直接文字列で指定する場合は▼Jnnnn▼（nnnnは職員番号）の形式でなければならない。

LRECL＝論理レコード長／$\left\{\begin{matrix}U\\E\\L\end{matrix}\right\}$

論理レコード長：　データプールの論理レコード長を正の整数または変数名で指定する。変数名で指定した場合はOPTION文のBUFFERパラメータを指定する必要がある。

$\left\{\begin{matrix}U\\E\\L\end{matrix}\right\}$：　入出力形式を指定する。U（FORMATなし），E（FORMAT付き），L（FORMAT付き／なし）のいずれか，または変数名を指定する。（現在はU以外の使用を制限している）。ここに書かれたU，E，LはFnn同様変数名とみなさない。

$\left\{\begin{matrix}\underline{\text{MAXRCD}}=\text{レコード数}\\ \text{TRK}=\text{トラック数}\end{matrix}\right\}$

データプールのサイズをレコード数またはトラック数で指定する。トラック数で指定する場合は変数名での指定をしてはならない。

MODE＝$\left\{\begin{matrix}\text{NEW}\\\underline{\text{OLD}}\end{matrix}\right\}$

NEW：　データプールを新規に創成するか，既に作成されているデータプールをクリアする場合に指定する。

OLD：　既に作成されているデータプールにデータの入出力等を行う場合に指定する。

NEW，OLDは変数名とみなさない。

OWNER＝所有者名

データプールの所有者を8文字以内の文字列または変数名で指定する。

DIRECT＝ディレクトリの個数

データプールがもつディレクトリの個数を指定する。ディレクトリにはノード名，一次元インデックスに関する情報や，入出力並び，注釈文などが格納される。データプールが必要とするディレクトリの個数は完成した樹状階層構造を想定し次の式で計算した値が目安となる。

ノードの数＋一次元インデックスの数＋（リークノード数＋一次元インデックスの数）*4

$+\sum_{i=1}^{n}((\text{コメント文の文字数})_i+15)/16$

（n：コメント文の数）

MACHINE＝$\left\{\begin{matrix}\text{▼F230-75▼}/\text{▼WORD▼}\\ \text{計算機名}/\left\{\begin{matrix}\text{▼WORD▼}\\\text{▼BYTE▼}\end{matrix}\right\}\end{matrix}\right\}$

データプールを作成する計算機名とワード・マシンかバイト・マシンかを指定する。計算機名は8文字以内の文字列で，文字定数か，変数名で指定する。

$$\text{DEVICE}=\left\{\frac{\text{▼F479B2▼}/135/13030}{\text{装置名}/\text{レコードギャップ}/1\text{トラックのバイト数}}\right\}$$

データプールを格納する大記憶装（ディスク）の装置名，1トラック当りの記録密度，レコードギャップを指定する。装置名は8文字以内の文字列で，文字定数か変数名で指定する。

4〕使用上の注意

FD文はパラメータで指定されたデータプールを使用するための初期設定を行う。しかしプログラム中にFD文がいくつ指定されていても，プログラムの実行時に実行されたFD文で指定されるデータプールの初期設定しか行わない。プリプロセッサはCJ文を展開するときFD文に対しては初期設定ルーチンの呼び出し文を作成すると同時にプログラム中で使用されているFD文の数やいくつかの情報を記憶しておく。プリプロセッサはプログラム全体の処理が済むとその情報をもとにしてプログラムの最後にデータプール・プロセッサの作業環境を設定するための1つのサブルーチンQMAINを指定により作成し付加する。このためデータプールを使用するプログラムには同時に使用すると考えられる最大のデータプール数だけFD文を定義するか，同時に使用するデータプールの数とFD文の数が一致しない場合はOPTION文のFILEパラメータでその数を指定する必要がある。

FD文は宣言文を除いて他のいかなるCJ文よりも前に実行されなければならない。

実行時に1つのプログラムの中で同一のデータプールを2度以上定義してはならない。また同一ファイル定義名でFD文を複数回実行させると最初の定義のみが優先し，2回目以降は無視する。

パラメータの指定において，データプールを新たに作成する場合（MODE＝NEWの場合）はDIRECTORYまでを必ず指定しなければならない。また既に作成されている場合（MODE＝OLDまたはこのパラメータを省略した場合）はMAXRCDまでを必ず指定しなければならない。それ以降は省略してもよい。

FD文は直接データプール・ファイルの割り当てを行っていない。したがってこのファイルはジョブ制御文のファイル割り当て文を用いて定義しておかなければならない。このときファイル定義名はFD文のファイル定義名に対応する名前をつけなければならない。M200計算機の場合は

F01 → FT01F001
F02 → FT02F001
⋮ ⋮ ⋮
F99 → FT99F001

5〕 使 用 例

(1)新たにデータプールを作成する。

データプール名 ： ▼J1648▼.▼DATAPOOL▼

ファイル定義名 ： F20

レコードサイズ ： 80

最大レコード数 ： 100

所 有 者 名 ： TOMIYAMA

ディレクトリの個数 ： 98

計算機名，装置名は標準値を用いる。

データプールの構成は最終的に図2.3で示すようになるものとする。

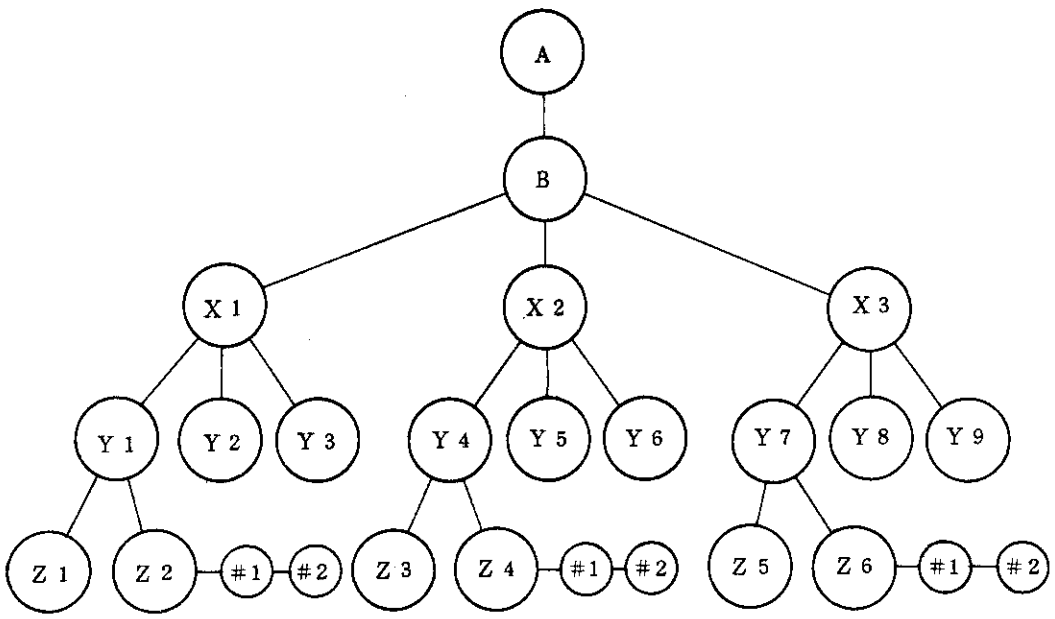


Fig. 2.3 A Type of Tree Structured File

```
CJ   FD  F20=▼J1648▼.▼DATAPOOL▼, LRECL=80/U, MAXRCD=
                 100, MODE=NEW,
     1  OWNER=▼TOMIYAMA▼, DIRECT=98
```

または

```
CJ   FD  F20=▼J1648▼, ▼DATAPOOL▼, 80/U, 100, NEW,
                 TOMIYAMA, 98
```

としてもよい。また各パラメータを実行時に指定する（例えば入力データとして与える）場合は

```
      DOUBLE PRECISJON A, B, OWNER
      INTEGER FNN, DIRECT
      READ(5, 10) FNN, A, B, LRECL, MAXRCD, MODE, OWNER,
                  DIRECT
   10 FORMAT(A3, 1X, A8, 1X, A8, 1X, I3, 1X, I5, 1X, A3, 1X,
              A8, 1X, I4)
CJ    FD FNN=A·B, LRECL/U, MAXRCD, MODE, OWNER, DIRECT
```

のように指定してもよい。

(2)既に作成されているデータプールの参照またはデータの更新追加登録などを行う例。

(1)で作成したデータプールの参照またはデータの更新追加登録などを行う。この場合MODE以下のパラメータが省略できる。ファイル定義名はF01とする。

```
CJ    FD F01=▼J1648▼.▼DATAPOOL▼, LRECL=80/U, MAXRCD
            =100
```

または

```
CJ    FD F01=▼J1648▼.▼DATAPOOL▼, 80/U, 100
```

としてもよい。また使用例3のようにすべてのパラメータを実行時に与えてもよい。

2.2.2 入出力文

2.2.2.1 WRITE文

1〕構　文

```
CJ   WRITE({データ名}〔, P=優先権〕〔, EXP=展開形式〕〔, FMT=書式名〕
           {*     }
           {#     }
           〔, ERR=文番号〕) 出力並び
```

2〕機　能

データプールにデータを格納する。格納されるデータのデータ名はパラメータで指定されたものかあるいはノード位置設定ポインタが示すものである。

3〕パラメータ

{データ名 / * / #}

データ名：　格納するデータにつける名前を指定する。

*　　：　ノード位置設定ポインタの示すノードにデータが格納される。

#　　：　一次元インデックス位置設定ポインタの示すノードにデータが格納される。

P＝優先権

格納するデータの優先権を0から9までの整数か変数名で指定する。格納しようとするデータのデータ名が既に登録されている場合には優先権どうしの比較を行う。もし新たに登録しようとするデータの優先権が既に登録されているものより等しいか大きければ，データは書き換えられる。また逆に小さければエラー・コンディションがONになり，ERRパラメータで

指定されている文番号をもつ文にジャンプする。ＥＲＲパラメータの指定がなければエラー・メッセージを出力しプログラムは停止する。

ＥＸＰ＝$\left\{\frac{0}{1}\right\}$

プリプロセッサによる展開形式を０，１または変数名で指定する。

０：正常に展開する。

１：ＷＲＩＴＥ文が生成されない。したがってこの場合は実行時にノード名だけが登録される。樹状階層構造を先に定義しておく場合などに使用する。

変数名：　ＷＲＩＴＥ文が論理ＩＦ文として生成される。

ＦＭＴ＝文番号

データプールへデータを書き出すときのFORMAT文の文番号を指定する。（現在この使用は制限している）。

ＥＲＲ＝文番号

データプール・アクセス・ルーチンがエラーを発見した場合，実行を継続する文の文番号を指定する。このパラメータが省略された場合，エラーが発見されるとプログラムは停止する。

出力並び

格納するデータの変数名を記述する。出力並びに添字付きの変数を書く場合，添字式は添字のつかない変数と整数の四則演算までしか許さない。またDO形並びの初期値，終値，増分値も同様である。

4〕使用例

(1)図 2.4 に示す階層構造のデータプールを新たに作成する。

ＦＤ文のパラメータ及びノード名Ａ，Ｂはすでに与えられているものとする。

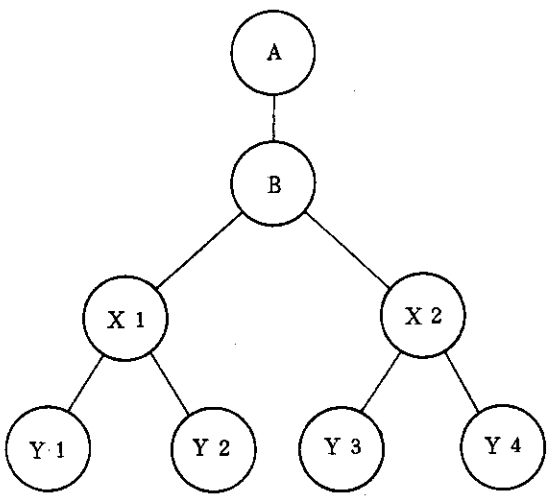


Fig. 2.4　A Datapool File Structure

```
CJ      FD  F01=A. B, LRECL/U, MAXRCD, MODE, OWNER, DIRECT
                              ⋮
CJ      WRITE(A・B・▼X1▼・▼Y1▼, ERR=1000)……
CJ      WRITE(A・B・▼X1▼・▼Y2▼, ERR=1000)……
CJ      WRITE(A・B・▼X2▼・▼Y3▼, ERR=2000)……
CJ      WRITE(A・B・▼X2▼・▼Y4▼, ERR=2000)……
```

(2)既に作成されているデータプールにデータの追加格納および既に格納されているデータの書きかえを行う。

(1)で作成したデータプールにA・B・▼X3▼・▼Y5▼, A・B・▼X3▼・▼Y6▼のデータ名でデータを追加する。またA・B・▼X2▼・▼Y3▼, A・B・▼X2▼・▼Y4▼のデータを書きかえる。

```
CJ      FD  F01=A・B, LRECL/U, MAXRCD
                              ⋮
CJ      WRITE(A・B・▼X3▼・▼Y5▼, ERR=1000)……
CJ      WRITE(A・B・▼X3▼・▼Y6▼, ERR=1000)……
CJ      WRITE(A・B・▼X2▼・▼Y3▼, ERR=2000)……
CJ      WRITE(A・B・▼X2▼・▼Y4▼, ERR=2000)……
```

樹状階層構造は図2.5に示すようになる。

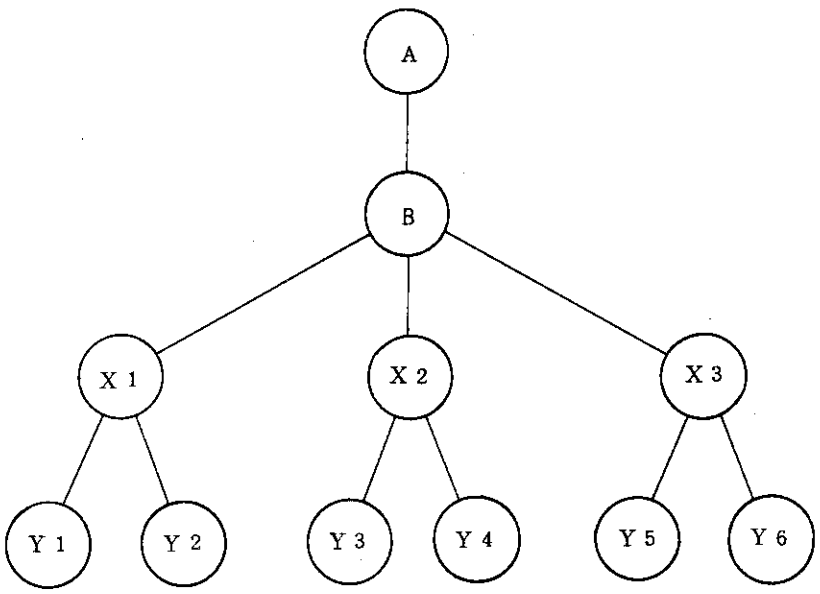


Fig. 2.5 Addition of Nodes

(3)データプールに階層構造のみを登録する。

樹状階層構造は図2.5で示されているものとする。また各ノード名は入力データで与えられるものとする。X, YはDOUBLE PRECISON の定義がされているものとする。

```
      DO  20  I=1, 6
      READ(5, 10)  X, Y
   10 FORMAT(8A, 1X, 8A)
CJ    WRITE('J1648'.'DATAPOOL'.X. Y, EXP=0, ERR=30)
   20 CONTINUE
```

(4)一次元インデックス付きデータの格納

図 2.4 に示すデータプールのデータ名A・B・X2・Y4に一次元インデックスつきのデータを格納する。一次元インデックスは#1, #5, #3 の順に与えるものとする。

```
CJ    WRITE(A・B・X2・Y4・#1, ERR=50)……
CJ    WRITE(A・B・X2・Y4・#5, ERR=50)……
CJ    WRITE(A・B・X2・Y4・#3, ERR=50)……
```

結果は図 2.6 に示すようになる。ここに一次元インデックスは昇べきの順に並んでいることに注意されたい。

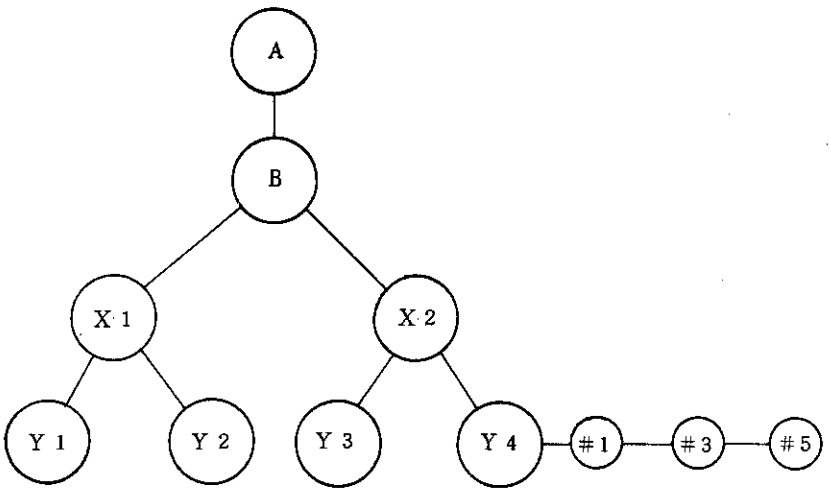


Fig. 2.6 Example of One Dimensional Index

(5)一次元インデックス付きデータの追加挿入

前例(4)で作成した一次元インデックス付のデータに#2と#4の一次元インデックス付データを追加挿入する。さらに#11から#20までを追加する。

```
CJ    WRITE(A・B・X2・Y4・#2, ERR=50)……
CJ    WRITE(A・B・X2・Y4・#4, ERR=50)……
      DO  10  N=11, 20
CJ    WRITE(A・B・X2・Y4・#N, ERR=50)……
   10 CONTINUE
```

結果は図 2.7 に示すようになる。

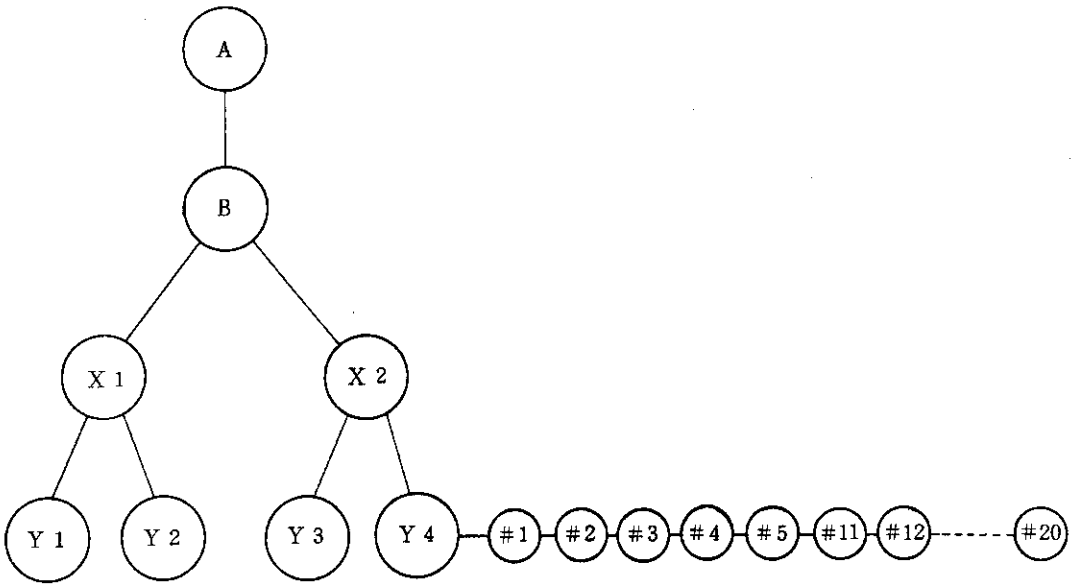


Fig. 2.7 Addition of One Dimensional Indices

(6)種々の深さのノードを扱う場合

データプール・プロセッサは指定されたデータ名の途中に空白のノード名があると空白の前までをデータ名とみなす。いま図 2.8 に示すような階層構造のデータプールにデータを格納する場合次のような方法もある。データ名と格納するデータは入力データとして与えられるものとする。

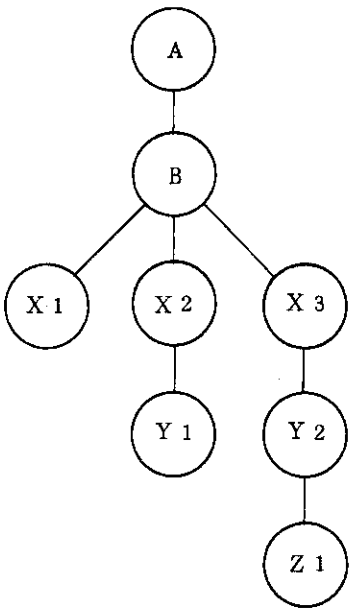


Fig. 2.8 Tree Structure with Different Depths of Node

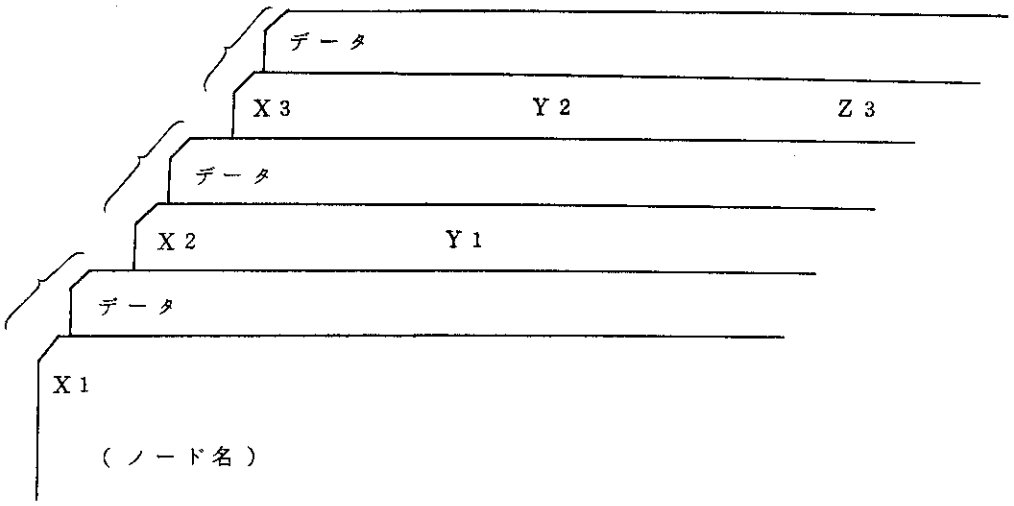


Fig. 2.9 Example of Input Data

```
      DOUBLE PRECISON A, B, X, Y, Z
                ⋮
      A=▼J1648▼
      B=▼DATAPOOL▼
      DO 30 I=1, 3
      READ(5, 10) X, Y, Z
   10 FORMAT(3(1X, A8))
      READ(5, 20) DATA
   20 FORMAT(……
CJ    WRITE(A・B・X・Y・Z, ERR=50) DATA
   30 CONTINUE
```

すなわち3種類のノードの深さを指定したWRITE文を使用しなくてもよい。データプールからデータを読む場合も同様である。

(7) *, #を用いたデータの格納

ノード位置設定文の項参照

2.2.2.2 READ文

1〕構　文

CJ　READ({ データ名 / * / # } [, ERR=文番号]) 入力並び

2〕機　能

指定されたデータ名をもつデータをデータプールから読む。

3〕 パラメータの説明

{データ名 / * / #}

データ名： 読み込むべきデータのもつデータ名を指定する。

*　　： ノード位置設定ポインタの示すデータ名のデータを読む。

#　　： 一次元インデックス位置設定ポインタの示すデータ名のデータを読む。

FMT＝文番号

書式付きでデータを読み込む場合のFORMAT文の文番号を指定する。（現在このパラメータの使用は制限されている）

ERR＝文番号

データプール・プロセッサがエラーを発見した場合に実行を継続する文の文番号を指定する。このパラメータを省略した場合，データ名が無い場合を除きプログラムを停止させる。データ名が無い場合プログラムは停止しないが強制的なデータの読取りが起りFORTRANエラーが出力された後ち，実行はこのCJ文の次の文に継続する。

4〕 使 用 例

(1)指定されたデータ名をもつデータを読む

図2.5で示されたデータプールからA・B・X1・Y1およびA・B・X2・Y4のもつデータを読む。

```
CJ      FD  F01=▼A▼・▼B▼, 80/U, 100
                          ⋮
CJ      READ(▼A▼・▼B▼・▼X1▼・▼Y1▼, ERR=50)……
CJ      READ(▼A▼・▼B▼・▼X2▼・▼Y4▼, ERR=50)……
```

または

```
        LRECL=80
        MAXRCD=100
        A=▼A▼
        B=▼B▼
        X1=▼X1▼
        X2=▼X2▼
        Y1=▼Y1▼
        Y4=▼Y4▼
CJ      FD  F01=A・B, LRECL/U, MAXRCD
CJ      READ(A・B・X1・Y1, ERR=50)……
CJ      READ(A・B・X2・Y4, ERR=50)……
```

でもよい。

(2)一次元インデックス付データを読む

図2.7で示されるデータプールから一次元インデックス付のデータを読む。読む順序は

#11から#20までを順番どうり，その後#5，#1，#3を読むこととする。

```
        DO 10 N=11,20
CJ      READ(A・B・X2・Y4・#N, ERR=50)……
   10   CONTINUE
CJ      READ(A・B・X2・Y4・#5, ERR=50)……
CJ      READ(A・B・X2・Y4・#3, ERR=50)……
CJ      READ(A・B・X2・Y4・#1, ERR=50)……
```

(3) *，#を用いたデータの読み込み

ノード位置設定文の項参照

2.2.2.3 COMMW文

1〕構　文

CJ　COMMW({データ名 / * / #}〔, P=優先権〕〔, ERR=文番号〕){▼文字列▼ / n, 変数名}

2〕機　能

指定されたノードに注釈文をつける。

3〕パラメータの説明

{データ名 / * / #}

データ名：　注釈文をつけるノードを最後に含むデータ名。データ名が存在しない場合は新たに登録し，そこに格納する。

*　：　ノード位置設定ポインタが示すノードに注釈文を格納する。

#　：　一次元インデックス位置設定ポインタが示すノードに注釈文を格納する。

ERR=文番号

データプール・プロセッサがエラーを発見した場合実行を継続すべき文の文番号を指定する。

{▼文字列▼ / n, 変数名}

格納すべきコメントを指定する。

▼文字列▼：　格納すべき注釈文を文字列で直接指定する場合の指定法．

n，変数名：　注釈文が格納されている変数名を指定する。指定する変数名には添字を付けてはならない。nは格納される注釈文の文字数．

4〕使用例

(1) A・B・C・Dで示されるデータ名のノードDに注釈をつける。COMMW文に直接注釈文を書く場合。

```
CJ   COMMW(A・B・C・D, ERR=50) ▼===COMMENT OF THIS DATA===▼
```

また注釈文がCOMという領域に格納されている場合は次のようにする。注釈文の文字数を30文字（空白を含む）とする。

M＝30

CJ　　COMMW(A・B・C・D, ERR＝50)　M, COM

2.2.2.4　COMMR文

1〕構　文

CJ　　COMMR({データ名 / * / #}〔, ERR＝文番号〕)　n, 変数名

2〕機　能

指定されたノードに格納されている注釈文を読む。

3〕パラメータの説明

{データ名 / * / #}

データ名：　読むべき注釈文の格納されているノード名を最後に含むデータ名。

*　　：　ノード位置設定ポインタが示すノードの注釈文を読む。

#　　：　一次元インデックス位置設定ポインタが示すノードの注釈を読む。

ERR＝文番号

データプール・プロセッサがエラーを発見した場合に実行を継続する文の文番号。

n：

読まれた注釈文の文字数が格納される変数名

変数名：

注釈文を読み込み，格納する領域の変数名

4〕使用例

(1) A・B・C・Dで示されるデータ名のノードDについている注釈文を読む。

COMという変数名の領域に注釈文を読み込む。注釈文の文字数はMに格納される。

```
        DIMENSION  COM(50)
                  :
CJ      COMMR(A・B・C・D, ERR＝50)  M, COM
```

2.2.2.5　SEARCH文

1〕構　文

CJ　　SEARCH(データ名，データ長〔, ERR＝文番号〕)

2〕機　能

データ名が存在するか否かをチェックする。データ名が存在する場合は変数名で指定された変数にデータ長を格納する。

3〕パラメータの説明

データ名：

存在するかどうかをチェックするデータ名を指定する。

データ長：

指定されたデータ名をもつデータのデータ長を格納する変数名を指定する。

ＥＲＲ＝文番号

データ名が存在しない場合，またはデータプール・プロセッサがエラーを発見した場合に実行を継続すべき文の文番号。

4〕使用例

(1)指定されたデータ名が存在するか否かをチェックする。

Ａ・Ｂというデータプールに Ａ・Ｂ・Ｃ・Ｄというデータ名があるかどうかをチェックする。存在しない場合文番号５０をもつ文から実行を継続する。

ＣＪ　　ＳＥＡＲＣＨ（Ａ・Ｂ・Ｃ・Ｄ，Ｌ，ＥＲＲ＝５０）

2.2.3　ノード位置設定文

既に述べたように樹状階層構造はデータ名で構成されている。データ名はいくつかのノードから構成され，それぞれ上下の関係をもっている。またあるデータ名の下に接続するノードは１つのグループを構成し横のつながりをもつ。ノードの深さが同じでも異なるデータ名の下に接続するノードはそれぞれ別のグループに属する。これら上下左右の関係はデータプール・プロセッサが樹状階層構造を構築していく段階で自動的に結びつけていく。これとは別にデータプール利用者がいくつかのノードを同一カテゴリに属するものとして結びつけることができる。たとえば図2.10に示すようにＡ・Ｂ・Ｘ1・Ｙ1，Ａ・Ｂ・Ｘ2・Ｙ5，Ａ・Ｂ・Ｘ3・Ｙ7，Ａ・Ｂ・Ｘ4・Ｙ10・Ｚ1を同一カテゴリとして結びつけたり逆に結びつきを解いたりすることがCONECT，DISCONECT文を用いて行える。CONECT文での結びつきをリレーショナル・ポインタ（relational pointen）による結びつきと呼ぶ。ノードの上下左右へのポインタ及び

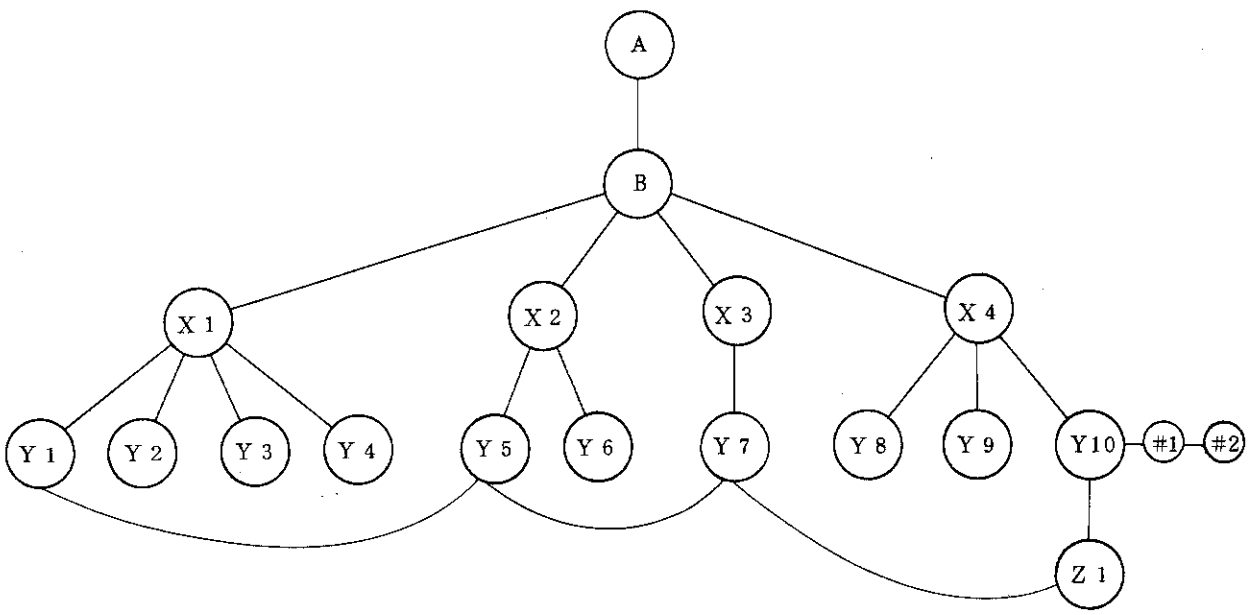


Fig. 2.10　Tree Structure Connected by Relational Pointers

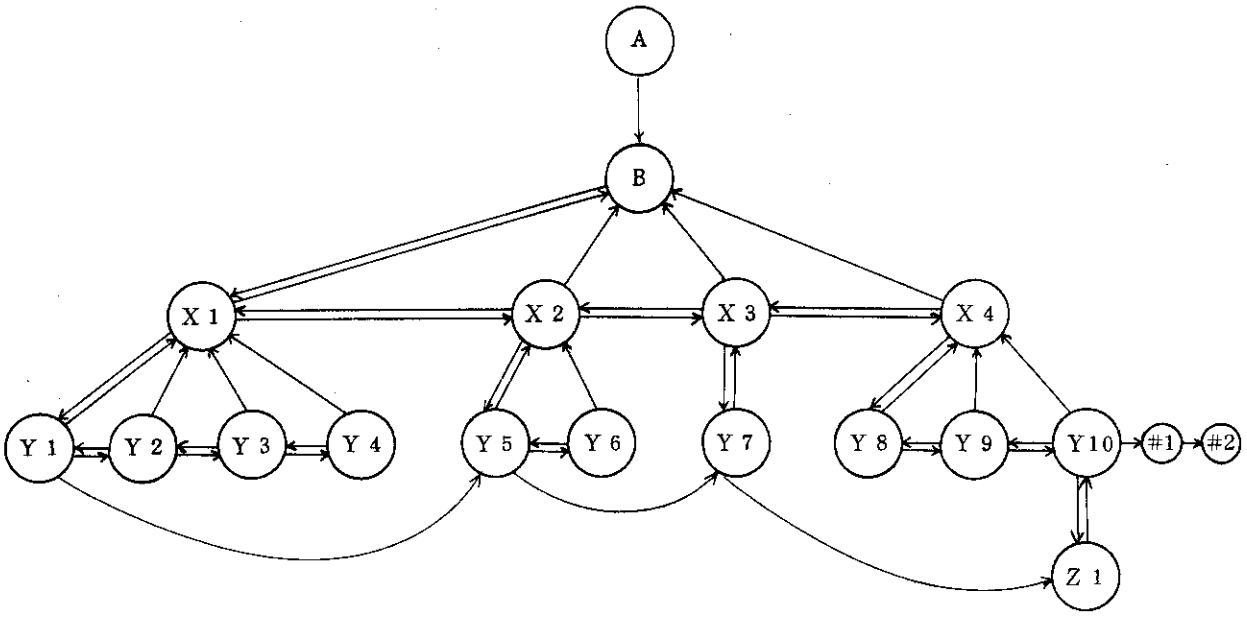


Fig. 2.11 Connections of Nodes

リレーショナルポインタはディレクトリに記録される。図2.10に示される樹状階層構造の各ノードの結びつきを概念的に示すと図2.11のようになる。

ノード位置設定文はノード名の探索や，特定のノードの結合などを行うものである。ノードの探索を行うためデータプール・プロセッサはノード位置設定ポインタ及び一次元インデックス位置設定ポインタと呼ばれるものを用意している。これらのものは現在処理の対象となっているノードがどれかを示すものである。ノード位置設定用ポインタは現在の位置からノード位置設定用のCJ文により図2.11に示す矢印の方向に移動させることができる。移動した先が新たに処理の対象となるノードである。一次元インデックス位置設定ポインタは一次元インデックス付データ名のみを扱う。位置設定ポインタの示すノードから逆にデータ名は一意に定まる。＊や＃でのデータの入出力はノード位置設定ポインタが示すデータのデータを入出力するものである。これらの位置設定ポインタは省略して位置設定ポインタあるいは単にポインタと呼ばれる場合がある。

2.2.3.1 POINT文

1〕構　文

CJ　POINT（データ名，D_{max}，D，V，RC）

2〕機　能

ノード名探索の開始点を指定する。D_{max}，D，V，RCにはこの文の実行時以外に他のノード位置設定用CJ文の実行によっても値が設定される。

3〕パラメータの説明

データ名

探索の開始点を指定する。これによりデータプール・プロセッサはノード位置設定用のポインタを指定されたデータ名の最後のノードに設定する。

D_{max}

樹状階層構造の最大の深さを正の整数または任意の整数型の変数名で指定する。

D

ノード位置設定ポインタが示しているノードの深さがデータブール・プロセッサにより格納される領域を整数型の任意の変数名で指定する。

V

ノード位置設定ポインタが示しているノードのデータ名をデータブール・プロセッサはユーザの指定した領域に格納する。データ名が格納される領域を任意の変数名で指定する。ノード名が2語単位で格納されるので領域はD_{max}だけ2語長（DOUBLE PRECISION）で確保しておく必要がある。POINT文が実行された時点ではVにはPOINT文で指定したデータ名が格納されている。

RC

リターン・コードが格納される領域を任意の整数型の変数名で指定する。ノード位置設定文の実行後この変数には次の値が格納される。

0： 正常終了

1： ノード名が存在しない。

4〕 使 用 例

(1)ノード名探索の開始点を指定する。

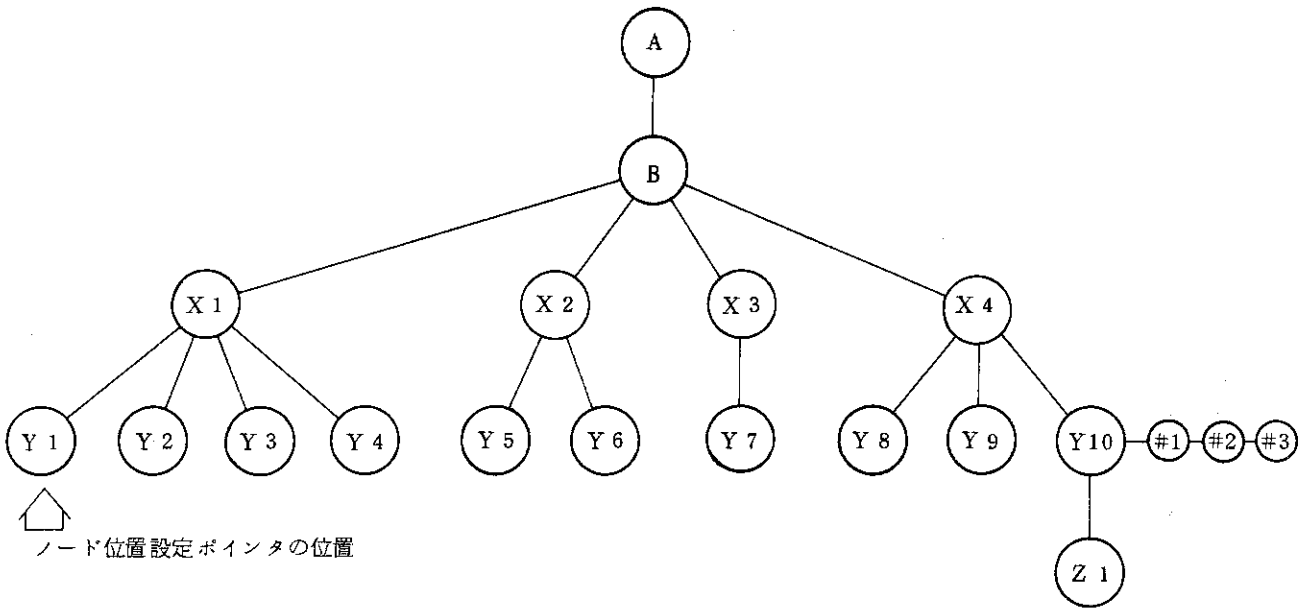


Fig. 2.12 Position of Pointer after Execution of POINT Statement

図2.12に示すデータプールにおいてノード名探索の開始点を指定する。

```
DOUBLE PRECISION VNAME(5)
DOUBLE PRECISION A, B
A=▼J1648▼
B=▼DATAPOOL▼
MAXD=5
CJ    POINT(A・B・▼X1▼・▼Y1▼, MAXD, ND, VNAME, NRC)
```

上記のPOINT文が実行されるとノード位置設定ポインタがA・B・X1・Y1の位置に設定される。またNDにはノードの深さ4が，VNAMEにはノード位置設置ポインタの示すノードのデータ名が

```
VNAME(1)=▼J1648▼
VNAME(2)=▼DATAPOOL▼
VNAME(3)=▼X1▼
VNAME(4)=▼Y1▼
```

の順で格納される。

(2)ノード位置設定ポインタの示すノードのデータを読む

```
CJ    POINT(A・B・▼X1▼・▼Y1▼, MAXD, ND, VNAME, NRC)
CJ    READ(*, ERR=50)……
```

2.2.3.2 NEXTH文

1〕構　文

```
CJ    NEXTH
```

2〕機　能

現在ノード位置設定ポインタで示されているノードと同じ深さで，右側に接続されているノードにノード位置設定ポインタを移動する。POINT文で指定されたノードの深さを示す任意の変数名Dの値は変らない。V(D)に新しいノード名が格納される。RCは0なら正常終了，1ならノードが存在しない。

3〕使用例

(1)ノード位置設定ポインタを右方向に順次移動する。図2.13で示されるデータプールにおいてA・B・X1・Y1と同じ深さのノード名をリストする。

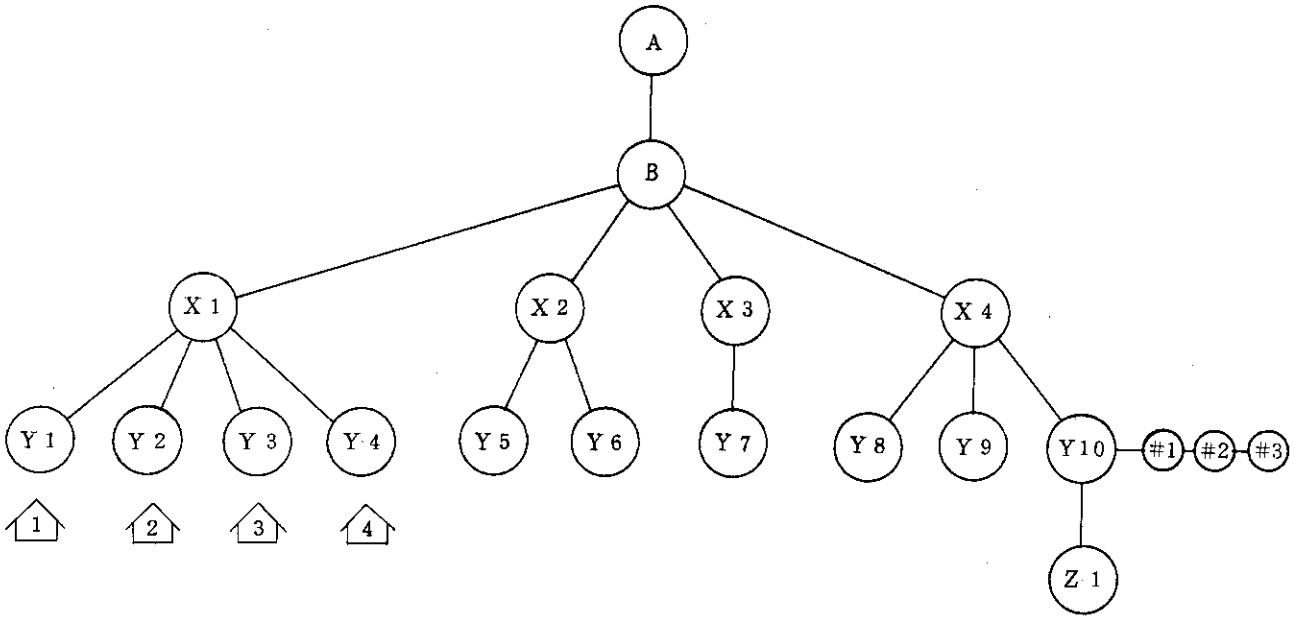


Fig. 2.13 Pointer Movement by NEXTH Statement

```
CJ      POINT(A·B·▼X1▼, ▼Y1▼, 5, ND, VNAME, NRC)
   100  IF(NRC·NE·0) GO TO 200
        WRITE(6, 10) VNAME(ND)
    10  FORMAT(1X, A8)
CJ      NEXTH
        GO TO 100
   200  CONTINUE
```

POINT文の実行によりノード位置設定ポインタは矢印の1が示す位置を指す。NEXTHを実行するとポインタは矢印の2の位置に移りVNAME(4)にY2が格納されている。ポインタが4の位置まで来た時点でNEXTHを実行するとNRCが≠0になりY4の右側にはX1の下に接続している単純ノードの無いことを示す。

(2)図2.13で示されるデータプールにおいてA·B·X1·Y1からA·B·X1·Y4までのデータを読む

```
CJ      POINT(A·B·▼X1▼·▼Y1▼, 5, ND, VNAME, NRC)
   100  IF(NRC·NE·0) GO TO 200
CJ      READ(*, ERR=200)……
              ⋮
CJ      NEXTH
        GO TO 100
   200  CONTINUE
```

2.2.3.3　NEXTV文

1〕構　文

```
CJ    NEXTV
```

2〕機　能

現在ノード位置設定ポインタで示されているノードに接続し，ノードの深さが1つ深い左端のノードにノード位置設定ポインタを移動する。POINT文で指定されたノードの深さを示す任意の変数名Dの値は1増加する。V(D)に新しいノード名が格納される。

3〕使用例

(1)ノード位置設定ポインタを下向に順次移動する。図2.14で示されるデータプールにおいてA・Bの下に接続するノードで最左端の単純ノード名をリストする。

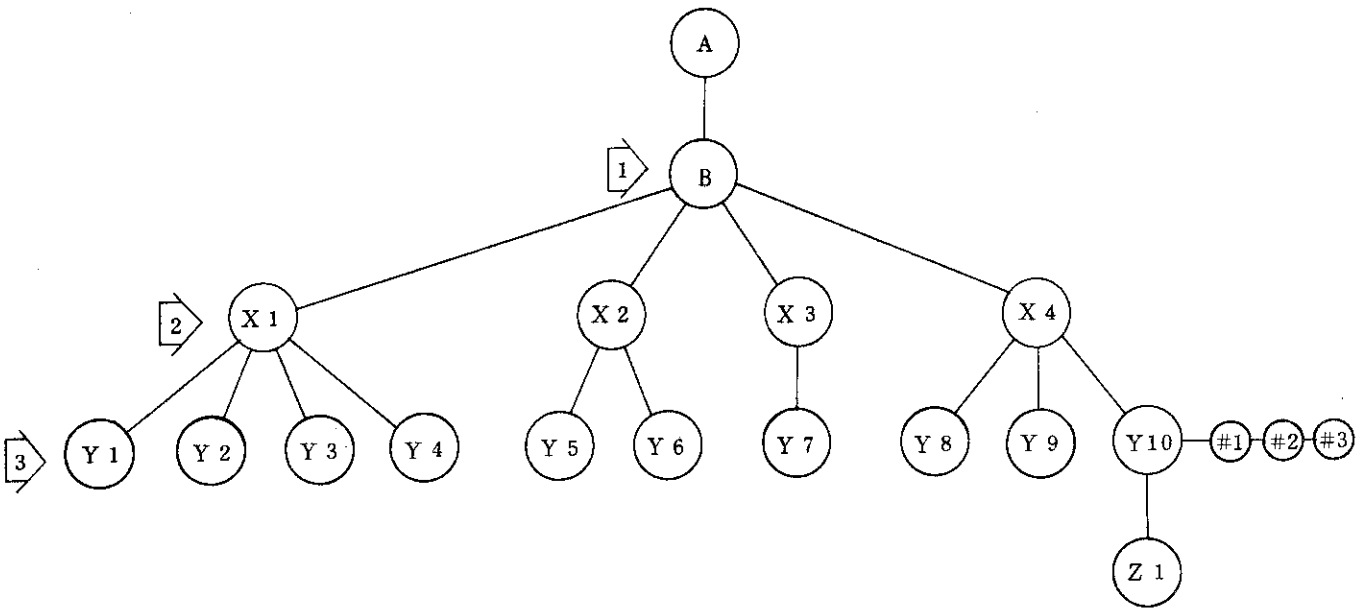


Fig. 2.14　Pointor Movement by NEXTV Statement

```
CJ     POINT(A・B, 5, ND, VNAME, NRC)
   100 IF(NRC・NE・0) GO TO 200
CJ     NEXTV
       GO TO 100
   200 WRITE(6, 10) VNAME(ND)
    10 FORMAT(1X, A8)
```

POINT文により矢印1の位置をノード位置設定ポインタが指す。NEXTVが実行されるとポインタは矢印2の位置に移り，NDには3が格納され，VNAME(3)には▼X1▼が格納される。次にNEXTV文が実行されるとポインタは矢印の3の位置に移

りNDには4が格納され，VNAME(4)には▼Y1▼が格納される。この時点でNEXTV文を実行するとNRCに1が格納されNDおよびVNAMEの値は変らない。

2.2.3.4　PRIORH文

1〕構　文

CJ　PRIORH

2〕機　能

現在ノード位置設定ポインタで示されているノードと同じレベルで左側に接続されているノードにノード位置設定ポインタを移動する。POINT文で指定されたノードの深さを示す任意の変数名Dの値は変らない。V(D)に新しいノード名が格納される。RCにはポインタの移動が正常に終了した場合0，ノードが存在しない場合1が格納される。

3〕使用例

(1)ノード位置設定ポインタを左方向に移動する。図2.15で示されるデータプールにおいてポインタが示すノードの左側のノードのデータを書きかえる。

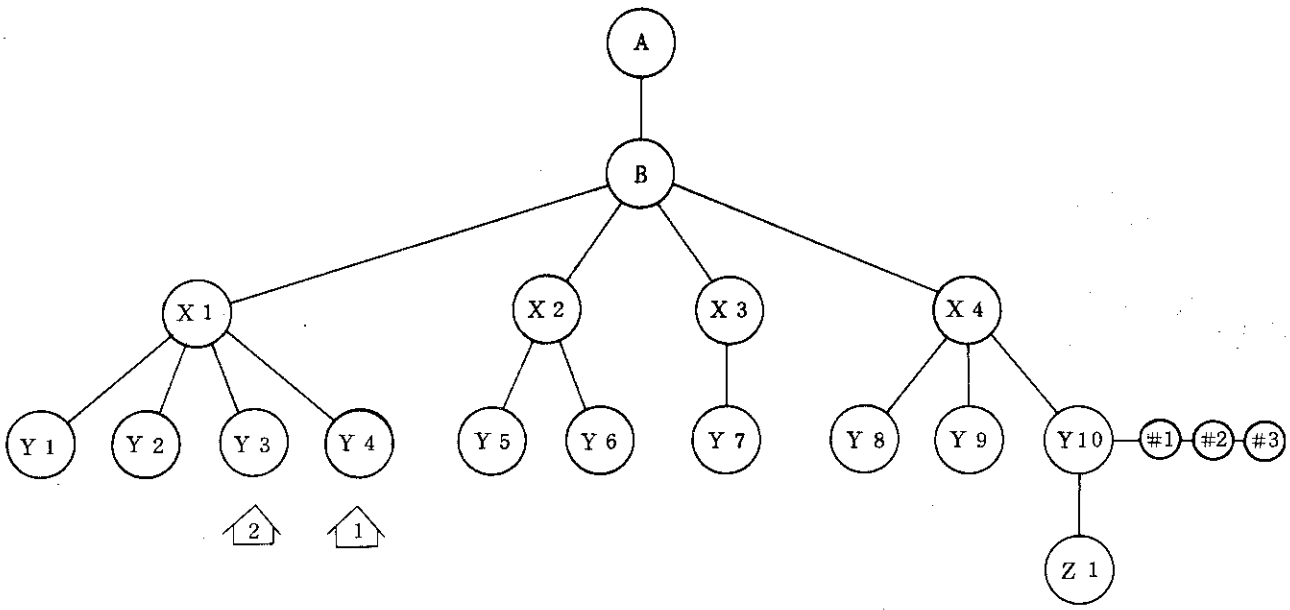


Fig. 2.15　Pointer Movement by PRIORH Statement

```
CJ      POINT(A・B・▼X1▼・▼Y4▼, 5, ND, VNAME, NRC)
CJ      PRIORH
CJ      WRITE(*, ERR=50) DATA
```

PRIORH文が実行されると矢印1で示されるノード位置設定ポインタの位置が矢印2で示される位置に移る。

2.2.3.5　PRIORV文

1〕構　文

```
CJ    PRIORV
```

2〕機　能

現在ノード位置設定ポインタで示されているノードに接続し，ノードの深さが1つ上のレベルのノードにポインタを移動する。POINT文で指定されたノードの深さを示す任意の変数名Dの値は1減じられる。V(D)に新しい単純ノード名が格納される。V(D+1)に格納されていたノード名は消去される。

3〕使用例

(1)ノード位置設定ポインタを上向に移動する。

図2.16で示されるデータプールにおいて矢印1が示すノード位置設定ポインタの位置を矢印2が示す位置に移動し，そのノード名をリストする。

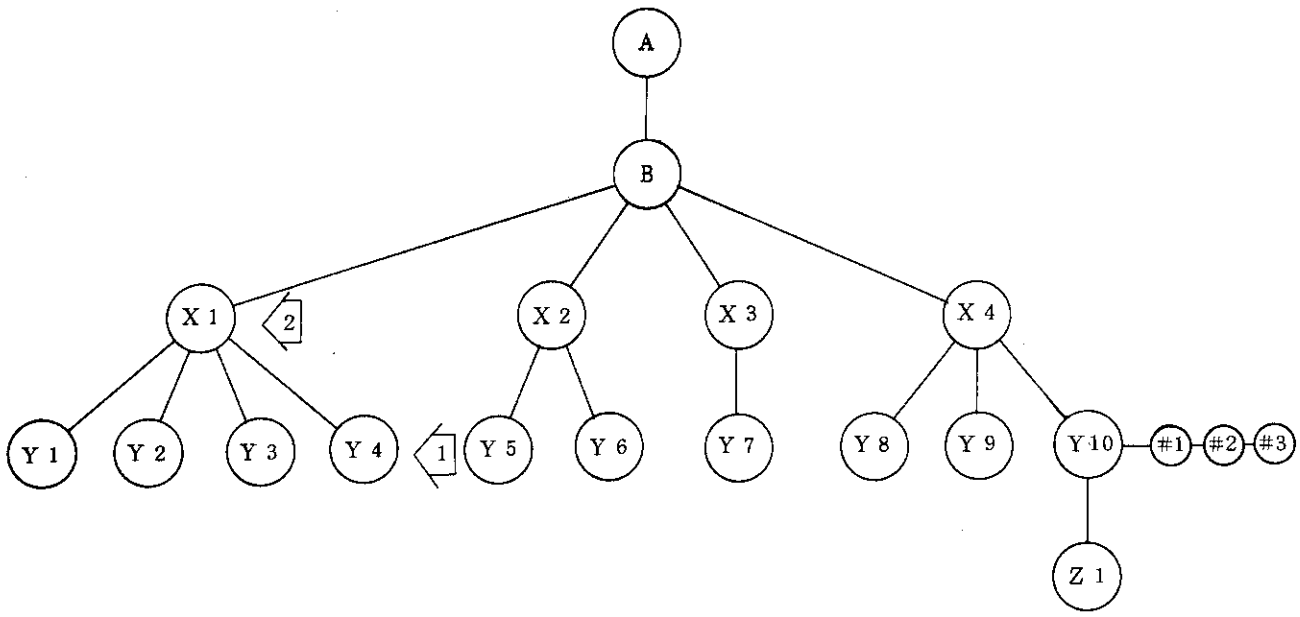


Fig. 2.16　Pointer Movement by PRIORV Statement

```
CJ      POINT(……, ND, VNAME, NRC)
                 ⋮
CJ      PRIORV
        WRITE(6, 10)  VNAME(ND)
   10   FORMAT(1X, A8)
```

2.2.3.6　CONNECT 文

1〕構　文

CJ　CONNECT（データ名1，データ名2，RC）

2〕機　能

データ名1，データ名2で示されたノードをリレーショナル・ポインタ（relational pointer）を用いて結合する。リレーショナル・ポインタはデータ名1からデータ名2へ向っている。結合はノードの深さが同一でなくてもよい。同一カテゴリに属するノードを結合する場合などに用いられる。

3〕パラメータの説明

データ名1

結合元となるデータの名前を指定する。

データ名2

結合されるデータの名前を指定する。

RC

リターン・コードを格納する任意の整数型変数名を指定する。CONNECT 文実行後この変数には次のうちいずれかの値が設定される。

0：　正常終了

1：　結合元となるデータ名が存在しない。

2：　結合先のデータ名が存在しない。

4〕使用例

(1)図 2.17 で示されるデータプールにおいて

A・B・X1・Y1

A・B・X2・Y5

A・B・X3・Y7

A・B・X4・Y10・Z1

をこの順にリレーショナルポインタを用いて結びつける。

CJ　CONNECT(A・B・X1・Y1, A・B・X2・Y5, NRC)

CJ　CONNECT(A・B・X2・Y5, A・B・X3・Y7, NRC)

CJ　CONNECT(A・B・X3・X7, A・B・X4・Y10・Z1, NRC)

結合は次の図 2.17 で示すようになる。

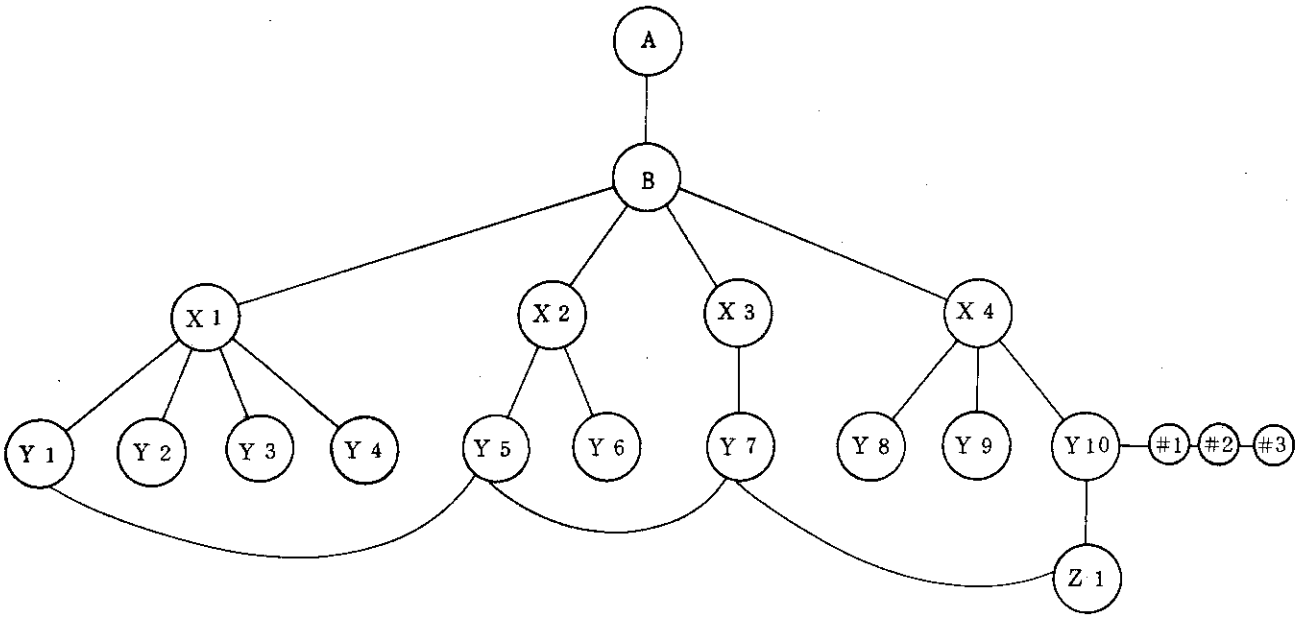


Fig. 2.17 Connection of Nodes by CONNECT Statement

2.2.3.7 DISCON文

1〕構　文

CJ　DISCON（データ名1，データ名2，RC）

2〕機　能

データ名1からデータ名2までリレーショナル・ポインタを順次たぐり，CONNECT文による結合を解く。

3〕パラメータの説明

データ名1，データ名2

結合を解くべき区間を指定する。データ名1で開始データ名を，データ名2で終了データ名を指定する。

RC

リターン・コードを格納する任意の整数型変数名を指定する。DISCON文実行後この変数には次のうちいずれかの値が設定される。

0：　正常終了

1：　開始データ名が存在しない。

2：　リレーショナル・ポインタが途中で切れている。

4〕使用例

(1)図2.17で示されるデータプールにおいて

A・B・X1・Y1とA・B・X2・Y5

の結びつきを解く

CJ　　DISCON(A・B・X1・Y1, A・B・X2・Y5, NRC)

(2) 2. 2. 3. 6.の使用例(1)で作成した結びつきを全て解く

CJ　　DISCON(A・B・X1・Y1, A・B・X4・Y10・Z1, NRC)

2. 2. 3. 8　RNEXT文

1〕構　文

CJ　　RNEXT

2〕機　能

現在ノード位置設定ポインタで示されているノードから，リレーショナル・ポインタで結合されているノードへ位置設定ポインタを移動する。この場合POINT文で定義されているDにはD_{max}と同じ値が格納され，新しい単純ノード名はV(D_{max})に格納される。

3〕使用例

(1) リレーショナル・ポインタをたぐりデータを読む。

図2.18で示されるデータプールにおいてA・B・X1・Y1から始まるリレーショナルポインタの結びつきを順次たぐりながらノード位置設定ポインタの示すノードのデータを読む。

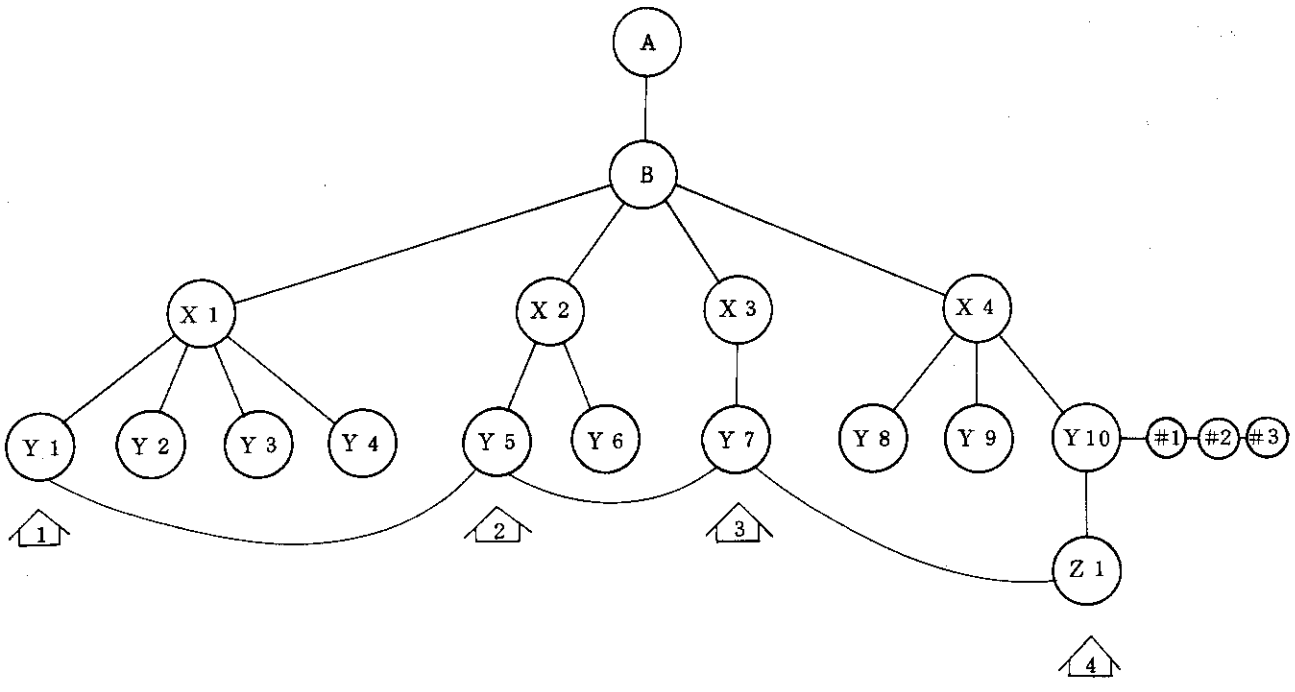


Fig. 2.18　Pointer Movement by RNEXT Statement

```
CJ      POINT(A・B・X1・Y1, 5, ND, VNAME, NRC)
   100  IF(NRC・NE・0) GO TO 200
CJ      READ(*, ERR=200)……
                ⋮
CJ      RNEXT
        GO TO 100
   200  CONTINUE
```

POINT文の実行によりノード位置設定ポインタは矢印1の位置に設定される。この時点ではNDには4が格納されVNAMEにはパラメータで指定したノード名が格納されている。RNEXT文が実行されるとノード位置設定ポインタは矢印2の位置に移る。この時点でデータプール・プロセッサはVNAMEにノード名の格納を行っていない。ただしNDにはD_{max}の値5を格納しVNAME(5)にノード名Y5が格納されている。データ名が知りたい場合はFIND文を用いなければならない。以後RNEXTを実行するとノード位置設定ポインタが順次移動する。ポインタが矢印4の位置にあるときRNEXT文を実行するとNRCに1が格納され，この後にリレーショナル・ポインタによる結合のないことを示す。

2.2.3.9　FIND文

1〕構　文

```
CJ      FIND
```

2〕機　能

ノード位置設定ポインタが指しているノードのデータ名を知る。データ名はPOINT文で定義したVに格納される。この文はRNEXT文を使用した場合用いられる。すなわちRNEXT文の実行時間を短かくするためデータプール・アクセス・ルーチンがVにノード名を格納していないためである。

3〕使用例

(1)図2.18に示されるデータプールにおいてA・B・X1・Y1から順次リレーショナル・ポインタをたぐりノード名をリストする。

```
CJ      POINT(A・B・X1・Y1, 5, ND, VNAME, NRC)
   100  IF(NRC・NE・0) GO TO 200
        WRITE(6, 10) VNAME(ND)
    10  FORMAT(1X, A8)
CJ      RNEXT
CJ      FIND
        GO TO 100
   200  CONTINUE
```

2.2.3.10　DNEXT文

1〕構　文

CJ　DNEXT（変数名）

2〕機　能

現在ノード位置設定ポインタで示されているノードに一次元インデックスがある場合その一次元インデックスの位置を探索する。一次元インデックスの位置は一次元インデックス位置設定ポインタで示される。一次元インデックス付きデータの入出力は一次元インデックス位置設定ポインタの示すノードに対して行われる。一次元インデックス位置設定ポインタの初期設定はPOINT文またはDNEXT文で行われる。

3〕パラメータの説明

変数名

一次元インデックス位置設定ポインタが示している一次元インデックスの内容（数値）が格納される変数，任意の整数型変数名を指定する。

4〕使用例

(1)図2.19に示されるデータプールにおいてA・B・X4・Y10の一次元インデックス付きデータを読む。

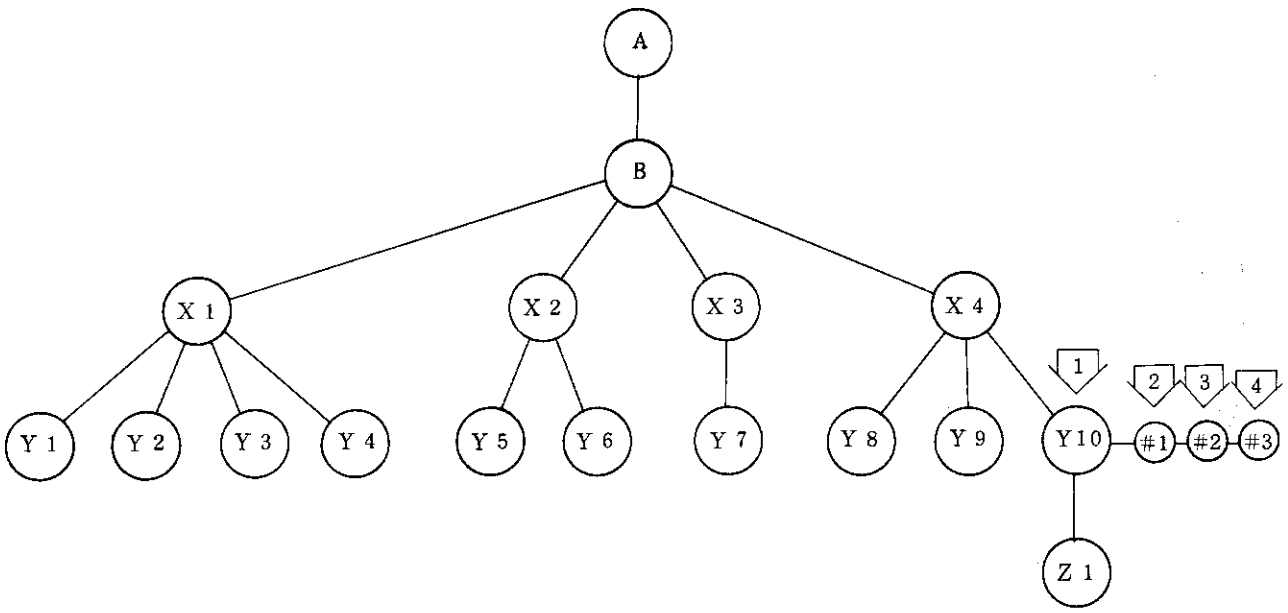


Fig. 2.19　Pointer Movement by DNEXT Statement

```
CJ      POINT(A・B・X4・Y10, 5, ND, VNAME, NRC)
  100 DNEXT(N)
      IF(NRC・NE・0)  GO TO  200
      READ(#, ERR=300)……
      GO TO 100
  200 CONTINUE
```

POINT文を実行することによりノード位置設定ポインタが矢印1の位置に設定される。この状態でDNEXT文を実行すると一次元インデックス付データ用の位置設定ポイントが#1の位置に設定され#によるデータの入出力が可能になる。以後DNEXT文を実行すると順次ポインタが移動しデータの入出力ができる。

2.2.4 保守管理文

2.2.4.1 CONDENSE文

1〕構　文

```
CJ    CONDENSE (データプール名, WF=nn)
```

2〕機　能

データプールのデータ領域はノードの消去などにより使用不可領域が増加する。CONDENSE文はこれらの領域を集め，使用可能領域とする。

3〕パラメータの説明

データプール名

コンデンスの対象となるデータプール名を指定する。

WF=nn

作業用ファイルの論理機番を指定する。nnは0から99までの整数か整数型の変数名を指定する。この作業用ファイルはユーザファイルを使用することが望ましい。

4〕使用例

(1)指定されたデータプール・ファイルの使用不可領域を使用可能領域とする。

```
        A=▼J1648▼
        B=▼DATAPOOL▼
CJ      FD F20=A・B, 80/U, 100
CJ      CONDENSE(A・B, WF=21)
```

F20としてデータプールのファイルを，また作業用ファイルとしてF21をジョブ制御文で割り当てておかなければならない。

2.2.4.2 SORT文

1〕構　文

```
CJ    SORT(データ名〔,LEVEL={i}〕〔,WF=nn〕〔,SR=m〕)
                            {ALL}
```

2〕機　能

指定されたノード名の下に接続するすべての樹状階層構造のうち，LEVELパラメータで指定された深さのノードをSRパラメータで指定した規則に従って配置を変更する。データ名の変更は起らない。

3〕パラメータの説明

データ名

このデータ名の下に継がる樹状階層構造が並べかえの対象となる。データプール名を指定

するとデータプール全体が対象となる。

LEVEL＝{ i / ALL }

i： 並べかえの対象となった樹状階層構造のうち並べかえを行うレベルを1以上の整数または整変数名で指定する。ALLは変数名とはみなさない。

ALL： 並べかえの対象となった樹状階層構造のすべてのレベルに対して並べかえを行う。

このパラメータを省略すると1を指定したものとみなす。

WF＝nn

並べかえのために使用する作業用ファイルの論理機番を2桁の正の整数または整変数名で指定する。作業用ファイルはユーザファイルを使用することが望ましい。

SR＝m

並べかえの規則を指定する。mは次の値のいずれかでこのパラメータを省略すると1とみなす。

m＝1 アルファベットの順に並べる

＝2 未定義

＝3 未定義

4〕使用例

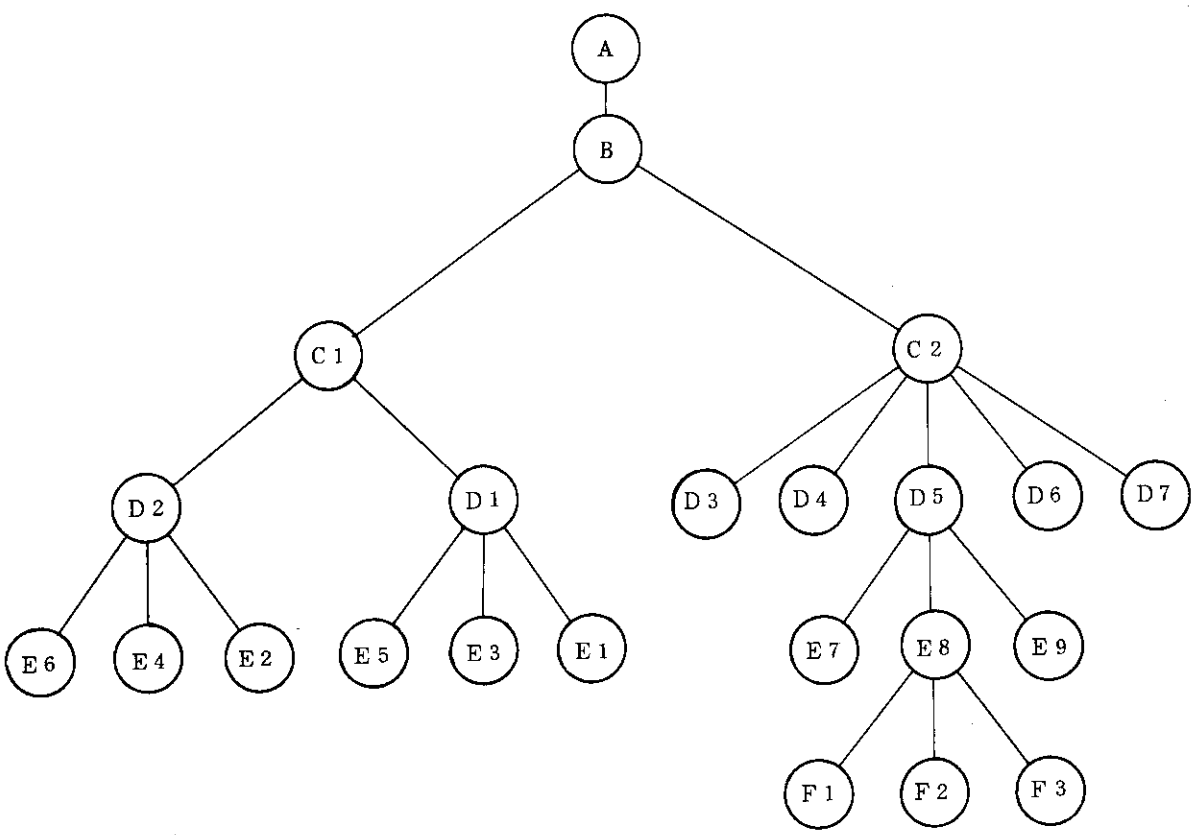


Fig. 2.20 Example of File for SORT Statement

(1)E5，E3，E1 を並べかえる

```
CJ      FD  F01=A・B，……
              ⋮
CJ      SORT(A・B・C1・D1，WF=10)
              ⋮
```

SORT文が実行されるとE5，E3，E1 が比較され，左からE1，E3，E5 の順に配置が変えられる。

(2)D2，D1を並べかえる

```
              ⋮
CJ      SORT(A・B・C1，WF=10
              ⋮
```

D2以下とD1以下が置きかわる。D2とD1のみの置きかえでないことに注意されたい。すなわちA・B・C1・D1・E6 というようなデータ名はできない。

(3)E6，E4，E2 およびE5，E3，E1 の並べかえ，すなわちA・B・C1に接続する2つの樹状階層構造の第2レベルを並べかえる。

```
              ⋮
CJ      SORT(A・B・C1，LEVEL=2，WF=12)
              ⋮
```

まずD2以下のE6，E4，E2 がE2，E4，E6 の順に並べかえられる。次にD1以下のE5，E3，E1 がE1，E3，E5 の順に並べかえられる。6つの単純ノードが左からE1，E2，…… E6 と並べられるようなことはない。

(4)A・B・C1以下に接続するすべての樹状階層構造のすべてのレベルに対して並べかえを行う。

```
              ⋮
CJ      SORT(A・B・C1，LEVEL=ALL，WF=10)
              ⋮
```

まずE6，E4，E2 の並べかえがなされ次にE5，E4，E1 の並べかえがなさせる。このレベルの並べかえがすべて終了すると1つ上のレベルのD2とD1の並べかえがなされる。

(5)データプール全体を並べかえる

```
              ⋮
CJ      SORT(A・B，LEVEL=ALL，WF=10)
              ⋮
```

2.2.4.3　RECOVER 文

1〕構　　文

```
CJ      RECOVER(データプール名 [,TYPE={W | {S, WF=nn} | {C, WF=nn}}])
```

2〕機　　能

データプールへデータの格納中，SORTやCONDENSE の実行中にCPUタイムオーバや何らかの原因による割り込み，マシンダウンなどが起りプログラムが強制的に終了させられるとデータプール・ファイルのクローズが正常に終了しないためデータプール管理テーブルが乱れ，以後データプールが使用できなくなる。RECOVER文はデータプール管理テーブルの乱れを修復しデータプールの使用を可能にするものである。

3〕パラメータの説明

データプール名

修復するデータプール名を指定する。

$$\text{TYPE}=\left\{\begin{array}{l}\underline{\text{W}}\\ \text{S, WF}=\text{nn}\\ \text{C, WF}=\text{nn}\end{array}\right\}$$

データプールが使用不可となった時行っていた作業は何かを指定する。

W：　データプールにデータを格納していた。

S：　SORT文が実行中であった。

C：　CONDENSE 文が実行中であった。

SまたはCを指定した場合はその時使用していた作業用ファイルを入力データ・ファイルとして指定する必要がある。nn は指定した入力データ・ファイルの論理機番を指定する。

4〕使 用 例

(1)データプールにデータを格納中CPUタイム・オーバ等の割込みが発生しプログラムの強制打切りが起った場合。

```
CJ      FD  F01=A・B, ……
CJ      RECOVER(A・B)
```

FD文で修復すべきデータプールを定義する。RECOVER文が実行されるとデータプールは修復される。

(2)SORT文の実行中にCPUタイム・オーバが起りプログラムの強制打切りが起った場合。

```
CJ      FD  F01=A・B, ……
CJ      RECOVER(A・B, TYPE=S, WF=02)
```

FD文で修復すべきデータプールを定義する。SORT文実行のために使用した作業用ファイルがジョブ制御文のファイル割り当て文で定義されておりその定義名がFT02F001とする。WFに論理機番として02を指定する。

2.2.5.　宣 言 文

2.2.5.1　OPTION文

1〕構　　文

```
CJ    OPTION 〔BUFFER＝作業領域サイズ／{U/L/E}〕
             〔,LRUSIZE＝{30/LRUテーブルサイズ}〕
             〔,MAXUDP＝最大使用データプール数〕
             〔,SAVE＝{1/テーブル保存間隔}〕
             〔,QMAIN＝{CREATE/NOCREATE}〕
```

2〕機　　能

データプール・アクセス・ルーチンの動作環境を設定する。この文が省略されるとプリプロセッサは標準値もしくはプログラム中に用いられているＣＪ文から各パラメータの値を決定する。通常は使用する必要はない。

3〕パラメータの説明

BUFFER＝作業領域サイズ／{U/L/E}

データプール・アクセス・ルーチンの使用する作業用領域のサイズと入出力形式をＵ，Ｌ，Ｅのいずれかで指定する（Ｌ，Ｅは現在使用を制限している）。このパラメータが省略されるとプリプロセッサはプログラム中のＦＤ文のＬＲＥＣＬパラメータから適当な値を決める。プログラム中にいくつものＦＤ文がありそのうち１つでもＬＲＥＣＬパラメータが変数名で指定されている場合は，このパラメータは必ず指定されなければならない。またプログラム中で定義されているＦＤ文の数と実行時に使用されるデータプールの数が一致しないときには指定の必要がある。データプールを２つ使用する場合の作業領域の構成を図 2.21 に示す。

PP_1	（13語）
PP_2	（13語）
$DPCTL_1$	（50語）
$DPCTL_2$	（50語）
$IOCTL_1$	（10語）
$IOCTL_2$	（10語）
LRUTBL	（LRUSIZE×6×25語）
I／O $BUFFER_1$	（$LRUTBL_1$語）
I／O $BUFFER_2$	（$LRUTBL_2$語）

Fig. 2.21　I／O Control Table of Datapool File

$$\text{LRUSIZE}=\left\{\begin{array}{l}30\\ \text{LRUテーブルサイズ}\end{array}\right\}$$

データプール・プロセッサは樹状層構造を構成するノードのディレクトリをLRUTBLテーブルにロードし使用している。このテーブルにロードされているディレクトリは不要とみなされると消去される。このテーブルはプログラムに1つ用意され，いくつかのデータプールで共用される。LRUテーブル・サイズはこのLRUTBLのサイズを変更したい場合に指定する。

MAXUDP＝最大データプール使用数

プログラム中で使用されているFD文の数と実行時に使用されるデータプールの数が一致しない場合，実行時に使用する最大データプール数を指定する。このパラメータが省略されるとプログラム中のFD文の数をその値として使用する。

$$\text{SAVE}=\left\{\begin{array}{l}1\\ \text{テーブル保存間隔}\end{array}\right\}$$

データの格納やノード名の登録を行うとデータプールを管理するテーブルが変更される。このテーブルはメモリ上のものとデータプールに格納されているものが一致している必要があり，標準的にはテーブルの変更があるたび保存している。SAVEパラメータはテーブルに変更があった場合に保存する間隔を変更する場合に指定する。また変更した場合はユーザ自身がプログラムの中のデータプールの使用が終了する位置にテーブルを保存する作業を行う文

```
CALL QSAVE(i)
```

を書いておかなければならない。

$$\text{QMAIN}=\left\{\begin{array}{l}\text{CREATE}\\ \text{NOCREATE}\end{array}\right\}$$

初期設定のためのサブルーチンQMAINをプリプロセッサが生成するか否かを指定する。

4〕使用例

```
CJ    OPTION  BUFFER=2000
CJ    FD  F01=A1・B1, LRECL=X1/U, MAXRCD=Y1
CJ    FD  F02=A2・B2, LRECL=X2/U, MAXRCD=Y2
CJ    FD  F03=A3・B3, LRECL=X3/U, MAXRCD=Y3
```

2.2.5.2　SYN文

1〕構　文

$$\text{CJ}\quad \text{SYN}\quad V_1=\left\{\begin{array}{l}\text{ノード名1}\\ \text{同義語1}\end{array}\right\}\cdot\left\{\begin{array}{l}\text{ノード名2}\\ \text{同義語2}\end{array}\right\}\cdots,\ V_2=\cdots$$

2〕機　能

同義語を宣言する。すなわちノード名の列を等号の左辺に書かれた変数名で代行させる。実行文中にデータ名を指定する場合，データ名またはその一部として同義語を指定することができる。同義語はまた他のSYN文が同義語を定義する際，ノード名の列の中に記述すること

ができる。プリプロセッサはＣＪ文を展開する際データ名の中に同義語が現われると，その部分を等号の右辺で置き換え展開する。

3〕 パラメータの説明

$$V_1 = \left\{ \begin{array}{l} \text{ノード名} \\ \text{同義語} \end{array} \right\} \cdots\cdots$$

同義語を定義するノード名または他の同義語をピリオドで区切って並べる。右辺のV_1は任意の変数名を指定する。この変数名が同義語として使用される。

4〕 使 用 例

```
CJ     SYN  S1=▼J1648▼.▼DATAPOOL▼, S2=A.B
CJ     SYN  S3=S1.S2, S4=X.S2.Y

CJ     SYN  S1=▼J1648▼.▼DATAPOOL▼
                  ⋮
CJ     READ(S1.X.Y, ERR=50)  DATA
```

2.2.5.3　ＡＬＩＡＳ文

1〕 構　　文

CJ　　ALIAS　　V_1 ＝予約語１〔，V_2 ＝予約語２，……〕

2〕 機　　能

プリプロセッサのもつ下記の予約語を変更する。

ＩＲＥＴＣＤ：　リターン・コードを格納する変数名

ＩＬＮＧＴＨ：　出力語数を格納する変数名

ＩＵＮＩＴ：　論理機番を格納する変数名

ＩＲＥＣＲＤ：　入出力データのレコード番号を格納する変数名

3〕 パラメータの説明

Ｖ＝予約語

Ｖは予約語の別名を任意の変数名で指定する。

4〕 使 用 例

(1)データプール利用プログラム中で既にＩＵＮＩＴという変数名が使われているので，プリプロセッサがＣＪ文の翻訳時にＩＵＮＩＴと出力する変数名はＩＵと出力するようにする。

```
CJ     ALIAS    IU=IUNIT
```

2.3 FORTRAN拡張文の展開規則

以下にプリプロセッサがどのようで形式にCJ文を展開するかを示す。展開規則の中でカナ，漢字，英小文字，イタリック体の英字はCJ文で与えられたパラメータからの情報が定数または変数名で出力されることを示す。また英大文字はそのまま出力される。

2.3.1 FD文

1〕 展開規則

```
CJ    FD Fnn=データプール名,LRECL=レコードサイズ/{U/F/L},{〔MAXRCD=〕最大レコード数 / TRK=トラック数}
        [,〔MODE=〕{OLD / NEW,〔OWNER=〕所有者名,〔DIRECT=〕
        ディレクトリの個数}[,〔MACHINE=〕{▼F230-75▼/▼WORD▼ / 計算機名/タイプ2},
        ,〔DEVICE=〕{▼F479B2▼/135/13030 / 装置名/レコードギャップ/1トラック当りのバイト数}]]
```

⇩

CALL　QTASK（Fnn，第1ノード名，N_1，第2ノード名，レコード長，タイプ$_1$，最大レコード数，モード，N_2，所有者名，ディレクトリの個数，N_3，計算機名，タイプ$_2$，N_4，装置名，レコードギャップ，バイト数）

N_1

第2ノード名が文字列の場合の文字数。第2ノード名が変数名の場合は0。

最大レコード数

トラック数が指定された場合はDEVICEパラメータから最大レコード数を決定する。

N_2

所有者名が文字列で指定された場合の文字数。変数名の場合は0。

N_3

計算機名が文字列で指定された場合の文字数。変数名の場合は0。

N_4

装置名が文字列で指定された場合の文字数。変数名で指定された場合は0。

2〕展開例

```
C=====================================================================
CJ     FD F01=USERN,DTPOOL,LRECL=80/U,MAXRCD=500,NO='NEW',
C     1     OWNER='TOMIYAMA',DIRECT=300
C=====================================================================
       CALL QMAIN
       CALL QTASK('F01',USERN,0,DTPOOL,80,'U',500,'NEW',8,'TOMIYAMA',300,
     18,'F230-75 ',0,2,8,'F47982  ',2,'PO',2,'DI',1,' ',1,' ',1,' ')
C=====================================================================
```

2.3.2 WRITE文

1〕展開規則

CJ　WRITE（{データ名 / * / #}〔, P＝優先権〕〔, EXP＝展開形式〕〔, FMT＝書式名〕〔, ERR＝文番号〕）出力並び

⇩

1) ILNGTH＝出力語数の計算式

2) CALL　QOPEN (I_{SN}, 2, K_{ey}, ILNGTH, IUNIT, IRECRD, P, m, t_1, n_1, v_{ar1}, t_2, n_2, v_{ar2}, … t_m, n_m, v_{arm}, N_1, S_{tm})

3) CALL　QWRITE (2, F_{mt}, L_{en}, ▼出力並び▼, j, k_1, d_{11}, d_{12}, … d_{17}, … k_j, d_{j1}, … d_{j7}, N_1, S_{tm})

4) WRITE (IUNIT▼IRECRD) 出力並び

5) CALL　QCLOSE (IRETCD)

I_{SN}

FORTRANコンパイラが1)につける内部発行の文番号。

2

WRITEオープンであることを示す。

k_{ey}

1，2，3のいずれか

1：　ノード名が指定されている場合

2：　＊が指定された場合

3：　＃が指定された場合

IUNIT

データプールが定義されている論理機番を格納する変数名。FD文で指定したFnnの値をデータプール・プロセッサが格納する。

m

指定されたノード名の深さ，＊または＃が指定された場合は0。

t

単純ノードの属性。

t ＝1　ノード名は変数名で指定されている。

　　2　ノード名は文字列で指定されている。

　　3　ノード名は一次元インデックスである。

n

単純ノードが文字列で指定された場合（ t ＝2 ）の文字数。他の場合は0。

v_{ar}

単純ノード名

N_1

ＥＲＲパラメータが指定されたか否かを示す。

N_1 ＝0　ＥＲＲパラメータは指定されていない。

　　1　ＥＲＲパラメータが指定されている。

S_{tm}

ＥＲＲパラメータで指定された文番号

F_{mt}

ＦＭＴパラメータで指定された文番号

L_{en}

出力並びの長さ（文字数）

j

以下に続くｋ，ｄの個数

k

出力並びに現われた変数名の属性を示す。４桁の整数でＩＩＪＫの形式で構成される。

ＩＩ：　変数名のタイプ

Table 2.2　Types of Variables

ＩＩ	意　　味
0	数値
1	文字定数
2	論理変数
3	半語長整数
4	整数
5	実数
6	倍精度実数
7	虚数
8	倍精度虚数
9	四倍精度実数
１０	四倍精度虚数
５０	サブルーチン名
６０	ファンクション名
７０	ステートメント番号

J： 変数がDimensionをもつ場合，変数の次元数で0～7の値をとる。

K： 0

d

Dimensionサイズ，Jで指定された数だけkに続いて現われる。

2〕展開例

```
C=====================================================================
CJ     WRITE(USERN,DTPOOL,'NODE1',#N,ERR=1234) DATA
C=====================================================================
       ILNGTH=1
       CALL QOPEN(3,2,ILNGTH,IUNIT,IRECRD,0,4,1,0,USERN,1,0,DTPOOL,2,5,'N
      1ODE1',3,0,N,1,&1234)
       CALL QWRITE(IRETCD,0,4,'DATA',1,500,1,&1234)
       WRITE(IUNIT'IRECRD,ERR=1234)DATA
       CALL QCLOSE(IRETCD)
C=====================================================================
```

2.3.3 READ文

1〕展開規則

CJ　READ({ データ名 / * / # } 〔, ERR＝文番号〕) 入力並び

⇩

1) CALL QOPEN (I_{sn}, 1, K_{ey}, ILNGTH, IUNIT, IRECRD, 0, m, t_1, n_1, v_{ar1}, t_2, n_2, v_{ar2}, … t_m, n_m, v_{arm}, N_1, S_{tm})

2) CALL QREAD (1, 0, L_{en}, ▼input list▼, j, k_1, d_{11}, … d_{17}, … k_j, d_{j1}, … d_{j7}, N_1, S_{tm})

3) READ (IUNIT▼IRECRD) input list

4) CALL QCLOSE (IRETCD)

パラメータの意味はWRITE文参照

2〕展開例

```
C=====================================================================
CJ     READ(USER,DTPOOL,AAAA,BBBB,#I) INPUT
C=====================================================================
       CALL QOPEN(8,1,ILNGTH,IUNIT,IRECRD,0,5,1,0,USER,1,0,DTPOOL,1,0,AAA
      1A,1,0,BBBB,3,0,I,0,0)
       READ(IUNIT'IRECRD)INPUT
       CALL QCLOSE(IRETCD)
C=====================================================================
```

2.3.4 COMMR／COMMW 文

1〕 展開規則

CJ COMMW ({ データ名 / * / # } 〔, P＝優先権〕〔, ERR＝文番号〕) { ▼文字列▼ / n , 変数名 }

CJ COMMR ({ データ名 / * / # } 〔, ERR＝文番号〕) n , 変数名

⇩

1) CALL QOPENC (I_{SN}, $\{^{1}_{2}\}$, K_{ey}, P, m, t_1, n_1, v_1, … t_m, n_m, v_m, N_1, S_{tm})

2) CALL QCOMMW／COMMR (IRETCD, $\{^{3}_{4}\}$, ILNGTH, { ▼文字列▼ / 変数名 }, N_1, S_{tm})

3) CALL QCLOSE

パラメータの意味はＷＲＩＴＥ文参照

2〕 展 開 例

```
C=====================================================================
CJ     COMMW(USERN,DTPOOL,NODE3,'NODE4',#1) 20,COMM
C=====================================================================
       CALL QOPENC(11,2,0,5,1,0,USERN,1,0,DTPOOL,1,0,NODE3,2,5,'NODE4',3,
      10,1,0,0)
       CALL QCOMMW(IRETCD,3,20,COMM,0,0)
       CALL QCLOSE(IRETCD)
C=====================================================================
```

2.3.5 SEARCH文

1〕 展開規則

CJ SEARCH (データ名, データ長〔, ERR＝文番号〕

⇩

1) CALL QOPEN (I_{SN}, 1, 1, ILNGTH, IUNIT, IRECRD, 0, m, t_1, n_1, v_{ar1}, … t_m, n_m, v_{arm}, N_1, S_{tm})

2) CALL QCLOSE (IRETCD)

2〕 展 開 例

```
C=====================================================================
CJ     SEARCH(USERN,DTPOOL,NODE3,NODE4,LENGTH,ERR=1234)
C=====================================================================
       CALL QOPEN(17,1,LENGTH,IUNIT,IRECRD,0,4,1,0,USERN,1,0,DTPOOL,1,0,N
      1ODE3,1,0,NODE4,1,&1234)
       CALL QCLOSE(IRETCD)
C=====================================================================
```

2.3.6 POINT文

1〕 展開規則

CJ　POINT（データ名，D_{max}，D，V，RC）

⇩

1) CALL　QPOINT（m，t_1，n_1，v_{ar1}，… t_m，n_m，v_{arm}，D_{max}，D，V，RC）

2〕 展 開 例

```
C=====================================================================
CJ     POINT(USERN,DTPOOL,NODE3,NODE4,MAXD,ND,VNAME,NRC)
C=====================================================================
       CALL QPOINT(4,1,0,USERN,1,0,DTPOOL,1,0,NODE3,1,0,NODE4,MAXD,ND,VNA
      1ME,NRC)
C=====================================================================
```

2.3.7 NEXTH/NEXTV/PRIORH/PRIORV/RNEXT

2〕 展開規則

CJ　NEXTH
CJ　NEXTV
CJ　PRIORH
CJ　PRIORV
CJ　RNEXT

⇩

1) CALL　QNEXT（i）

i = 1　NEXTH

2　NEXTV

3　PRIORH

4　PRIORV

5　RNEXT

2〕 展 開 例

```
C=====================================================================
CJ     NEXTH
C=====================================================================
       CALL QNEXT(1)
C=====================================================================
```

2.3.8 DNEXT文

1〕 展開規則

CJ　DNEXT（変数名）

⇩

1) CALL　QDNEXT（変数名）

2〕展 開 例

```
C=====================================================================
CJ     DNEXT(N)
C=====================================================================
       CALL QDNEXT(N)
C=====================================================================
```

2.3.9 CONNECT 文

1〕展開規則

CJ　　CONNECT（データ名1，データ名2，RC）

⇩

1) CALL　QCONEC（m_1，t_{11}，n_{11}，v_{ar11}，…，t_{1m1}，n_{1m1}，v_{ar1m1}，m_2，t_{21}，n_{21}，v_{ar21}，…t_{2m2}，n_{2m2}，v_{ar2m2}，RC）

2〕展 開 例

```
C=====================================================================
CJ     CONNECT(USERN,DTPOOL,NODE3,NODE4,USERN,DTPOOL,AAAAA,BBBBB,NRC)
C=====================================================================
       CALL QCONEC(4,1,0,USERN,1,0,DTPOOL,1,0,NODE3,1,0,NODE4,4,1,0,USERN
      1,1,0,DTPOOL,1,0,AAAAA,1,0,BBBBB,NRC)
C=====================================================================
```

2.3.10 DISCON文

1〕展開規則

CJ　　DISCON（データ名1，データ名2，RC）

⇩

1) CALL　QDISCO（m_1，t_{11}，n_{11}，v_{ar11}，…t_{1m1}，v_{ar1m1}，m_2，t_{21}，n_{21}，v_{ar21}，…t_{2m2}，n_{2m2}，v_{ar2m2}，RC）

2〕展 開 例

```
C=====================================================================
CJ     DISCON(USERN,DTPOOL,NODE3,NODE4,USERN,DTPOOL,AAAAA,BBBBB,NRC)
C=====================================================================
       CALL QDISCO(4,1,0,USERN,1,0,DTPOOL,1,0,NODE3,1,0,NODE4,4,1,0,USERN
      1,1,0,DTPOOL,1,0,AAAAA,1,0,BBBBB,NRC)
C=====================================================================
```

2.3.11 FIND文

1〕展開規則

CJ　　FIND

⇩

1) CALL　FIND

2〕展 開 例

```
C=================================================================
CJ     FIND
C=================================================================
       CALL @FIND
C=================================================================
```

2.3.12 CONDENSE 文

1〕展開規則

CJ　　CONDENSE（データプール名，WF＝nn）

⇩

1) CALL　QCOND（2, t_1, n_1, v_{ar1}, t_2, n_2, v_{ar2}, nn）

2〕展 開 例

```
C=================================================================
CJ     CONDENSE(USERN,DTPOOL,WF=NN)
C=================================================================
       CALL @COND(2,1,0,USERN,1,0,DTPOOL,NN)
C=================================================================
```

2.3.13 SORT 文

1〕展開規則

CJ　　SORT（データ名〔，LEVEL＝{ i / ALL }〕〔，WF＝nn〕〔，SR＝m〕）

⇩

1) CALL　QSORT（m, t_1, n_1, v_{ar1}, …, t_m, n_m, v_{arm}, i, SR）

　　i：　レベル

　　SR：　ソートの規則番号

2〕展 開 例

```
C=================================================================
CJ     SORT(USERN,DTPOOL,NODE3,LEVEL=ALL,WF=NN)
C=================================================================
       CALL @SORT(3,1,0,USERN,1,0,DTPOOL,1,0,NODE3,0,NN)
C=================================================================
```

2.3.14 RECOVER 文

1〕 展開規則

CJ　RECOVER（データプール名 [, TYPE = { W / { S, WF = nn / C, WF = nn } }]）

⇩

1) CALL　QRECOV（2, t_1, n_1, v_{ar1}, t_2, n_2, v_{ar2}, t, nn）

t：　修復のタイプ

nn：　WF の値

2〕 展 開 例

```
C=====================================================================
CJ     RECOVER(USERN,DTPOOL,TYPE=W)
C=====================================================================
       CALL QRECOV(2,1,0,USERN,1,0,DTPOOL,'W',0)
C=====================================================================
```

3. ＴＳＳ用データプール処理システム

3.1 ＴＳＳ用データプール処理システム（ＤＰＴＳ）について

端末からデータプールを利用するシステムを紹介する。システムに与えるインプットは後述するコマンドである。システムは図 3.1 のように

ＤＰＴＳ起動
Manual モード　（端末からインプット）
Run モード　（POOLPROC からインプット）
テキストモード　（端末からインプット）
パラメータモード（端末からインプット）
ＤＰＴＳ終了

の６状態を取るコマンドの形式は，コマンド名とオペランドに別れている。また，PROCEDURE とＥＮＤコマンドを用いてプログラムの作成が可能である。利用者は，各ユーザのファイル POOLPROC に格納し，ＲＵＮコマンドで起動する。

コマンドは次の２つの形式で１から７２カラム内に記述する。

コマンド名＋空白＋オペランド
文番号＋空白＋コマンド名＋空白＋オペランド

継続行は２ケ以上の空白＋ハイフンをオペランドに付け足す。変数名，定数は８文字以内，文番号は４桁以内に記述する。表 3.1 にコマンド類別表を，表 3.2 にコマンド一覧表を示す。

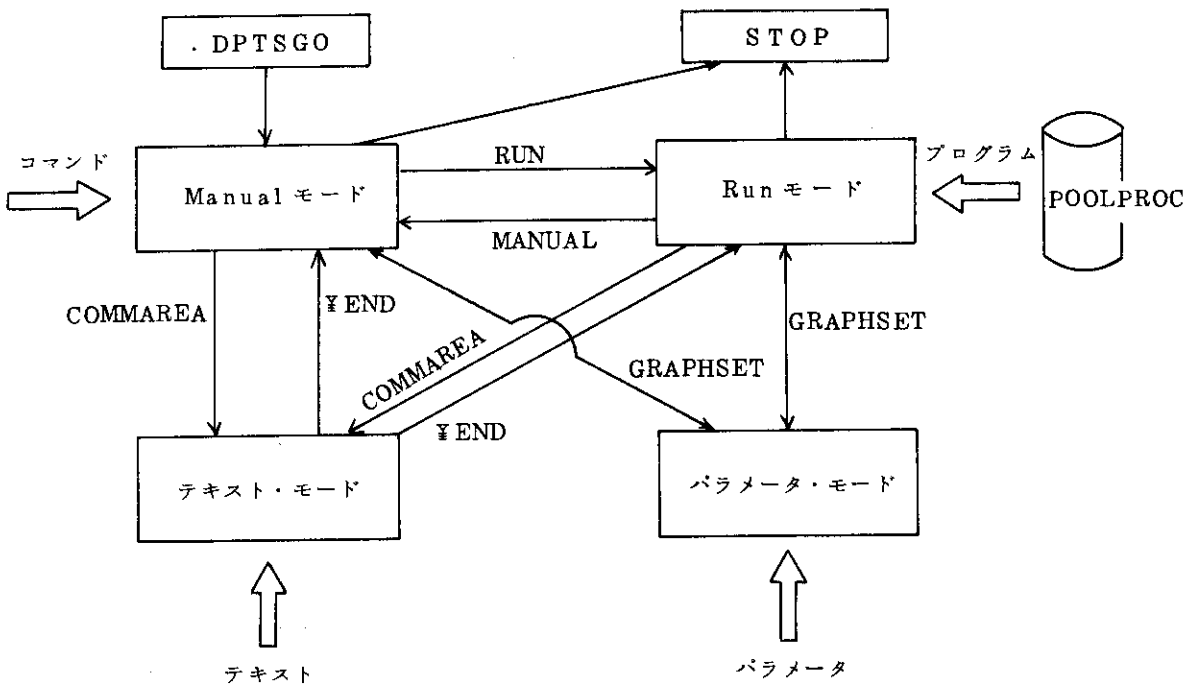


Fig. 3.1　State Transition of DPTS System

Table 3.1 DPTS Commands

機　　能	コマンド名
プログラム制御	STOP PROCEDURE MANUAL GOTO END RUN
データプール割付	ATTACH ALLOCATE
代　入　文	SYNONYM 代入文
データプール参照	CATLIST LIST LISTD
データプール操作	COMPARE MOVE COPY DELETE RENAME CHAP
条　件　文	ON
テキスト操作	COMMAREA DISPLAY CEDIT
コメント操作	COMMR COMMW COMMWA COMMD
グラフィック操作	GRAPHX GRAPHY GRAPHZ GRAPH GRAPHSET

Table 3.2 Command Table

コマンド名 〔コマンド名省略形〕	オペランド	説明	ページ
STOP	〔▼メッセージ▼〕	DPS実行終了	
PROCEDURE 〔PROC〕	プログラム名	プログラム定義文	
MANUAL	〔▼メッセージ▼〕	Manual モードへ戻る	
GOTO	文番号	GOTO文	
END		プログラム定義の終り	
RUN	プログラム名〔▼L▼〕	プログラム実行文	
ATTACH 〔A〕	ファイル定義名 {,/=} データブール名 , LRECL=nnn〔/{L/U}〕, MAXRCD=mmm	データブールの割り付け	
ALLOCATE 〔ALLOC〕	ファイル定義名 {,/=} データブール名 , LRECL=nnn〔/{L/U}〕, MAXRCD=mmm , MAXDT=KKK	データブールの作成	
SYNONYM	変数名=ノード名〔.ノード名 〔.〜〕〕	同義語定義文	
変数名=	▼文字列▼ {{変数名/定数}〔{+/−/*/}{変数名/定数}〕}	代入文	
CATLIST 〔CATL〕	データブール名	データブール情報の印刷	
LIST	データ名〔/{C, F, P, pRn}〕	データ情報の印刷	
LISTD 〔LD〕	データ名〔/変数名 〔, 初期値, 終値, 増分値〕〕	数値データの印刷	
COMPARE	データ名, データ名	比較	
MOVE	データ名, データ名	移動	
COPY	データ名, データ名	複写	
DELETE 〔DELE〕	データ名	消去	

コマンド名 〔コマンド名省略形〕	オペランド	説明	ページ
RENAME 〔R〕	データ名，データ名	改称	
CHAP	データ名 [/{ALL, A} / {ONLY, O}], {m, +m, −m} 〔, pRn〕	プライオリティの変更	
ON	ERROR {GOTO 文, STOP 文, MANUAL文}	エラー処置	
ON	{EQUAL, UNEQUAL} {GOTO 文, STOP 文, MANUAL文}	比較後の処置	
ON	({変数名, 定数}・Re・{変数名, 定数}) {GOTO 文, STOP 文, MANUAL文}	条件文	
COMMAREA 〔CAREA〕	n，A〔(m)〕	テキストの読み込み	
テキスト		コメント・テキスト	
¥END		テキスト終了文	
DISPLAY 〔DISP〕	〔n，A〔(m)〕〕	テキスト表示	
CEDIT	▼▼文字列▼▼文字列▼▼	文字列の置換	
COMMR 〔CR〕	データ名，変数名〔，A(m)〕	コメントの読み込み	
COMMW 〔CW〕	データ名，n〔，A(m)〕	コメントの書き込み	
COMMWA 〔CWA〕	データ名，n〔，A(m)〕	コメントの追加書込み	
COMMD 〔CD〕	データ名	コメントの消去	
GRAPHX 〔GX〕	データ名〔／変数名 〔，初期値，終値，増分値〔，T〕〕〕	xデータの読み込み	
GRAPHY 〔GY〕	同　上	yデータの読み込み	
GRAPHZ 〔GZ〕	同　上	zデータの読み込み	

コマンド名 〔コマンド名の省略形〕	オペランド	説明	ページ
GRAPH 〔G〕	OPEN 〔O〕	グラフィック・オープン	
同上	2D 〔省略〕	2次元図形のプロット	
同上	ADD 〔A〕	重ね図	
同上	ADDA	重ね図，Y軸を追加	
同上	3D	3次元図形のプロット	
同上	ERASE 〔E〕	画面消去，パラメータ・リセット	
同上	CLOSE 〔C〕	グラフィック・クローズ	
GRAPHSET 〔GS〕	LBLX 〔LX〕	x軸ラベルの読み込み	
同上	LBLY 〔LY〕	y軸ラベルの読み込み	
同上	LBLZ 〔LZ〕	z軸ラベルの読み込み	
同上	TITLE 〔TL〕	図のタイトル読み込み	
同上	LIMS 〔LIM〕	xmin，xmax，ymin，ymax を読む	
同上	SIZE 〔SIZ〕	x軸，y軸の分割数読み込み	
同上	LOG	軸スケールの読み込み	
同上	3D	3次元パラメータの読み込み	

3.2 コマンド

各コマンドの記述形式，機能，使用例を以下に示す。

3.2.1 STOPコマンド

記述形式

STOP　　〔▼文字列▼〕

機能

DPTSプログラムを終了する。

使用例

DPTSプログラムを開始し，終了する。

```
#. DPTSGO
>STOP
  *** DPS終了メッセージ ***
#
```

3.2.2 PROCEDURE コマンド

記述形式

PROCEDURE　プログラム名

〔PROC〕

機能

PROCEDURE 文はプログラムの開始位置を示し，そのプログラム名を与える。

オペランドの説明

プログラム名

英字で始まる8文字以内の英数字列。

使用例

プログラムPTESTを作成する。

```
#SWPR POOLPROC
#NEW
#10 PROCEDURE PTEST
    ...
#90 END
#SAVE PTEST
```

3.2.3 MANUALコマンド

記述形式

MANUAL　　〔▼文字列▼〕

機能

Runモードから Manual モードへ移行する。

オペランドの説明

MANUALコマンド実行時にこの文字列を出力する。

使用例

プログラムPTEST内で使用する。

```
    PROCEDURE PTEST
        ...
    ON ERROR GOTO 100
    MANUAL  ▼NORMAL RETURN▼
100 MANUAL  ▼ABNORMAL RETURN▼
    END
```

3.2.4 GOTOコマンド

記述形式

GOTO　　文番号

機能

Runモード下で実行され，プログラムの実行を指定された文番号をもつ文に移す。

オペランドの説明

文番号

4桁以内の正定数である事。

使用上の注意

GOTO と記述する事。GOとTOの間に空白を含まない事。

使用例

ループを作成する。

```
    PROCEDURE PTEST
    SYN S01=j0000·DPOOL·#N
    N=0
100 N=N+1
    ON(N·EQ·10) MANUAL
      ...
    GOTO 100
    END
```

3.2.5 ENDコマンド

記述形式

END

機能

プログラム定義の終りを示す。

使用例

プログラムＰＴＥＳＴを定義する。

```
PROCEDURE PTEST
    ...
    ...
END
```

3.2.6 ＲＵＮコマンド

記述形式

ＲＵＮ　　プログラム名〔,▼Ｌ▼〕

機能

Manual モードでＲＵＮコマンドを発し，ＰＯＯＬＰＲＯＣに格納されているプログラムを実行する。この時ＤＰＴＳはRunモードに移行する。

オペランドの説明

プログラム名

ＰＯＯＬＰＲＯＣ内にあるプログラムの名前。

▼Ｌ▼

Ｌが指定されると，実行中のコマンドをリストする。

使用上の注意

プログラム内のコマンド記述に誤りがある場合，またはプログラム内で文番号の使用方法に誤りがある場合，このプログラムは実行されず，Manual モードのままである。

使用例

プログラムＰＴＥＳＴを実行する。

```
#. DPSRUN PROC=PTEST
    ...
>RUN PTEST,▼L▼
    ...
```

3.2.7 ＡＴＴＡＣＨコマンド

記述形式

ＡＴＴＡＣＨ　　ファイル定義名 {,/=} データプール名

〔Ａ〕

,ＬＲＥＣＬ＝nnn〔／{Ｌ/Ｕ}〕,ＭＡＸＲＣＤ＝mmm

機能

既に作成されているデータプールを割付ける。

オペランドの説明

ファイル定義名

F05, F06, F07, F98, F99以外のファイル参照番号とする。

データプール名

SYNONYMで定義された変数名でも良い。

nnn

レコード長。単位はLの場合はバイト，Uの場合は語である。

mmm

最大レコード数

使用例

```
...
SYNONYM  S01=J9131・DPOOL
ATTACH   F02, J9131・TEST, LRECL=80/U, MAXRCD=500
A        F01=S01, LRECL=800, MAXRCD=2000
...
```

3.2.8 ALLOCATEコマンド

記述形式

```
ALLOCATE     ファイル定義名 {,/=} データプール名
〔ALLO〕      ,LRECL=nnn〔/{L/U}〕,MAXRCD=mmm
             ,MAXDT=kkk
```

機能

新規にデータプールを作成する。

オペランドの説明

ファイル定義名

F05, F06, F07, F98, F99 以外のファイル参照番号とする。

データプール名

SYNONYMで定義された変数名でも良い。

nnn

レコード長。単位はLの場合はバイト，Uの場合は語である。

mmm

最大レコード数

KKK

最大ダイレクトリ数

使用例

```
ALLO  F01, J9131・DPOOL, LRECL=80/U, MAXRCD=500,
            , MAXDT=200
```

3.2.9 SYNONYMコマンド

記述形式

SYNONYM　　変数名＝ノード名〔，ノード名〔，～〕〕

〔SYN〕

機能

データプール内のノードを表現する同義語定義文。

オペランドの説明

変数名

SYNONYM変数名

ノード名

8文字以内の文字列または変数名，一次元インデックス。

使用上の注意

SYNONYM変数名と変数の間に同一の変数名の使用は許されない。右辺のノード名の並びの最大個数は20個である。

使用例

SYN　X1＝J9999・DATAPOOL・A・B・C・#N

SYN　X2＝J9999・DATAPOOL

SYN　X3＝A・B・C

と定義すればデータX1とX2・X3・#Nは同一のデータを指す。

3.2.10 代入文

記述形式

$$\text{変数名}=\left\{\begin{array}{l}\left\{\begin{array}{c}\text{変数名}\\ \text{定数}\end{array}\right\}\left[\left\{\begin{array}{c}+\\ -\\ *\\ /\end{array}\right\}\left\{\begin{array}{c}\text{変数名}\\ \text{定数}\end{array}\right\}\right]\\ \blacktriangledown\text{文字列}\blacktriangledown\end{array}\right\}$$

機能

左辺の変数に値または文字列を代入する。

オペランドの説明

文字列

引用符でくくられた8文字以内の文字列である。

使用例

A＝10

B＝▼BBB▼

C＝A

D＝A／3

算術式は整数型に評価されるのでDには3が代入される。

3.2.11 CATLISTコマンド

記述形式

CATLIST　　データプール名

〔CATL〕

機能

データプール内の以下の情報を印刷する。

最大レコード数

使用レコード数

最大ダイレクトリ数

使用ダイレクトリ数

作成年月日

レコード長

オペランドの説明

データプール名

SYNONYMで定義された変数名でも良い。

使用上の注意

CATLISTコマンド以前に，ATTACHまたはALLOCATEコマンドによりデータプールが定義されている事。

使用例

```
...
SYN      S01=J9131・POOL
ATTACH   F01=S01, LRECL=80/U, MAXRCD=1000
CATLIST      S01
...
```

3.2.12 LISTコマンド

記述形式

LIST　　データ名〔/{C, F, P, pRn}〕

機能

データプール内の各データの持つ情報を出力する。

オペランドの説明

データ名

データプール名のみの指定であれば全てのデータに対して処理を行う。第3ノード以上の指定であればそのデータに対してのみ処理を行う。

C ：　コメントの出力を指示する。

F ：　入出力形式の出力を指示する。

P ：　プライオリティの出力を指示する。

pRn ：　ダイレクトリの優先度pと正整数nが関係Rを満たすとき，そのダイレクトリの

指示された項目を出力する。関係RはGT，GE，EQ，LE，NE のひとつである。

使用例

最初に全データ名を出力し，次に特定のデータの入出力形式を出力する。

```
...
LIST J9999・TESTPOOL
...
SYNONYM SSS=J9999・TESTPOOL・ABC
LIST SSS/F
...
```

3.2.13 LISTDコマンド

記述形式

LISTD　　データ名〔／変数名〔，初期値，終値，増分値〕〕

〔LD〕

機能

データの持つ数値データを出力する。

オペランドの説明

データ名

数値データの存在するデータを指定する。

変数名

数値データの持つ変数名を指定する。

初期値，終値，増分値

数値データ出力時の制御変数。

／以下のオペランドが省略された場合全てのデータが出力される。

使用例

データがS01・ABCでその入出力形式が（XX(I)，YY(I)，I=1，10）である場合。

LISTD	S01・ABC/XX	:	変数XXの値が出力される。
LD	S01・ABC/YY	:	変数YYの値が出力される。
LD	S01・ABC	:	XX，YYの値が共に出力される。
LD	S01・ABC/XX，1，10，3	:	XX(1)，XX(4)，XX(7) XX(10)の値が出力される。

使用上の注意

DPTS内のバッファ長が6000語であるので，6000語を超えるデータは処理できない。また出力行を少なくするために1回の出力量は1000語を超えてできない。

3.2.14 COMPAREコマンド

記述形式

COMPARE
〔COMP〕 { データプール名1，データプール名2
　　データ名1　　，データ名2 }

機能

データプール内の各データの持つ数値データ，コメント，入出力形式，プライオリティを比較する。

オペランドの説明

データプール名が指定されたら，それぞれのデータプール内の全データについて比較を行う。

データ名の指定，つまり第3ノード以上の指定，ならばそのデータどうしの比較を行う。

使用上の注意

データの持つ数値データ長が6000語を超える場合は比較できない。

使用例

COMPARE　J9131・DPOOL，J9131・WORK

COMP　J9131・DPOOL・ABC，J9131・WORK・DEF

3.2.15 MOVEコマンド

記述形式

MOVE　　データ名1，データ名2

機能

データにつながる数値データ，コメント，入出力形式の情報を移動する。

オペランドの説明

データ名1

転送元データであり，移動後消去される。

データ名2

転送先データであり，既存ならば重ね書き，無ければ新規作成する。

使用上の注意

データの持つ数値データ長が6000語を超える場合は移動できない。

使用例

MOVE　J9131・DPOOL・ABC，J9131，DPOOL，CDF

3.2.16 COPYコマンド

記述形式

COPY　{ データプール名1，データプール名2
　　データ名1　　，データ名2 }

機能

データにつながる数値データ，コメント，入出力形式の情報を複写する。

オペランドの説明

データプール名1またはデータ名1

転送元データであり，複写後消去されない。

データプール名2またはデータ名2

転送先データであり，既存ならば重ね書き，無ければ新規作成する。

データプール名が指定されたら，全データを複写する。データ名が指定されたらそのデータのみを複写する。

使用上の注意

データの持つ数値データ長が6000語を超える場合は複写できない。

使用例

COPY　J0013・DPOOL, J0124・DPOOL

COPY　J0650・DPS・ABC, J0124・DPOOL・ADD

3.2.17 DELETE コマンド

記述形式

DELETE　　データ名

〔DELE〕

機能

指定されたデータ以下につながるデータを消去する。

オペランドの説明

データ名

消去されるデータ名，データプール名の指定は不可能。

使用上の注意

消去作業はそのデータ名を▼????????▼に置換する事である。実際にデータプール内から消去されるのはコンデンス作業を行った時である。またデータプール名を指定してCOPYコマンドを使用した際，デリートされたデータは複写の対象にならない。

使用例

DELETE　J0880・DPOOL・ABC

DELETE　J0880・DPOOL・JJ・#9

3.2.18 RENAMEコマンド

記述形式

RENAME　　データ名1，データ名2

〔R〕

機能

データ名を改名する。

オペランドの説明

データ名1

旧名

データ名2

新名

使用例

▼ABC▼を▼CDE▼に改名する。

RENAME　S01・ABC, S01・CDF

3.2.19 CHAP (Change Priority) コマンド

記述形式

CHAP　データ名 [/{ALL, A} / {ONLY, O}], {m, +m, −m} [, pRn]

機能

データのプライオリティを変更する。

オペランドの説明

データ名

ALL, A ： データ名で示されるデータ以下の全てのリーブデータを対象とする。

ONLY, O ： データ名で示されるデータのみを対象とする。

m ： プライオリティをmにする。mは1桁の正数。

+m ： 〃 にmを加える。

−m ： 〃 からmを減ずる。

pR ： LT, LE, GT, GE, EQ, NE のいずれか。

n ： 1桁の正数

各データのプライオリティとnとでpRnの関係を調べ真ならプライオリティの変更を行い，偽なら行わない。

使用上の注意

プライオリティにmを加えて9より大の場合は9とする。

プライオリティからmを減じて0より小の場合は0とする。

使用例

CHAP　S01・ABC/ALL, +5, LE4

3.2.20 ONコマンド (1)

記述形式

ON　ERROR {GOTO 文 / STOP 文 / MANUAL 文}

機能

障害発生時に右辺のコマンドを実行する。

オペランドの説明

GOTO, STOP, MANUAL

それぞれのコマンドに同じ。

使用例

```
PROCEDURE PTEST
SYN S01=J0013・DATALIB
A    F20=S01, LRECL=556/U, MAXRCD=2000
ON ERROR STOP ▼ATTACH ERROR▼
MANUAL    ▼ATTACH NORMAL▼
END
```

3.2.21 ONコマンド (2)

記述形式

$$\text{ON}\ \left\{\begin{array}{l}\text{EQUAL}\\ \text{UNEQUAL}\end{array}\right\}\ \left\{\begin{array}{l}\text{GOTO}\quad\text{文}\\ \text{STOP}\quad\text{文}\\ \text{MANUAL文}\end{array}\right\}$$

機能

COMPAREコマンドによるノードの比較後の処置を行う。

オペランドの説明

第1オペランド

EQUAL　：　一致であれば第2オペランドを実行する。

UNEQUAL：　不一致であれば第2オペランドを実行する。

第2オペランド

GOTO, STOP, MANUAL

それぞれのコマンドに同じ

使用例

```
    …
COMPARE  S01・▼ABC▼, S02・▼CDF▼
ON EQUAL GOTO 10
COPY S01・▼ABC▼, S02・▼CDF▼
10  …
```

3.2.22 NOコマンド (3)

記述形式

```
ON  ( { 変数名 } { . EQ . } { 変数名 } { GOTO    文 }
      { 定数   }   { . NE . }   { 定数 }   { STOP    文 }
                   { . GT . }             { MANUAL 文 }
                   { . GE . }
                   { . LT . }
                   { . LE . }
```

機能

条件文である。

オペランドの説明

第1オペランド

かっこ内の論理式が真ならば第2オペランドを実行する。

第2オペランド

GOTO, STOP, MANUAL

それぞれのコマンドに同じ。

使用例

プログラム内でループ処理を行う。

```
PROC   RENAME
       ...
       N=0
10     N=N+1
       M=N*10
       RENAME   S01·#N, S01·#M
       ON (N·LT·10) GOTO 10
       MANUAL   ▼RENAME END▼
       END
```

3.2.23 COMMAREAコマンド

記述形式

COMMAREA n, A〔(m)〕

〔CAREA〕

機能

端末からコメント・テキストを読み込む。

オペランドの説明

n : これから読み込むコメントの文字数を指定する。定数又は変数で表現可。

m : コメントを読み込む領域Aの読み込み開始位置を指定する。定数又は変数で表現可。

使用例

CAREA　80, A(1)　：　Aの1語目から最大80文字分コメントを読み込む。

テキストを打ち込む（一行72文字まで）

¥END　：　テキストの終了を意味する。

DISPLAY　：　テキストをプリントする。

CE ▼▼文字列1▼▼文字列▼▼　：　テキスト内の文字列の置換。

COMMW　S01　：　テキストをデータS01に書き込む。

3.2.24 DISPLAYコマンド

記述形式

DISPLAY　〔n, A〔(m)〕〕

〔DISP〕

機能

コメントテキストを端末へ出力する。

オペランドの説明

n　：　出力するテキストの文字数

m　：　出力するテキストの領域Aの開始位置。

オペランドが省略されたら，直前に扱われたテキストが出力される。

使用例

…

CR　S01, N, A(M)　：　データS01からコメントを読み込む。

DISPLAY　：　読み込んだコメントを出力する。

…

3.2.25 CEDITコマンド

記述形式

CEDIT　▼▼文字列1▼▼文字列2▼▼

〔CE〕

機能

テキスト内の文字列の置換

使用例

…

CR　S01, N, A(I)　：　データS01からコメントを読む。

DISP　：　端末に表示する。

CE ▼▼STAT▼▼STATE▼▼　：　文字列の置換

CW　S01　：　修正されたコメントを書く。

…

3.2.26 COMMRコマンド

記述形式

COMMR　　データ名，変数名〔，A(m)〕

〔CR〕

機能

データからコメント・テキストを読む。

オペランドの説明

データ名

コメントの存在するデータの指定。

変数名

読み込まれたコメント・テキストの文字数が設定される変数を指定。

m

コメントを読み込む領域Aの読み込み開始位置を指定する。

使用例

```
...
CR    S01, N, A(1)      :  データS01からコメントを読む。
DISP      N, A(1)       :  端末に表示する。
...
```

3.2.27 COMMWコマンド

記述形式

COMMW　　データ名，n〔，A(m)〕

〔CW〕

機能

データへコメント・テキストを書く。

オペランドの説明

データ名

コメントを書くデータの指定

n　：　書き込むコメントの文字数を指定。

m　：　領域Aからの開始位置。

使用例

```
...
CAREA   80, A(1)        :  Aの1語目から最大80文字分コメントを読み込む。
テキストを打ち込む
¥END                    :  テキストの終了。
COMMW   S01             :  データS01へ書く。
...
```

3.2.28 COMMWAコマンド

記述形式

COMMWA　　データ名，n〔，A(m)〕

〔CWA〕

機能

データへコメント，テキストを追加する。

オペランドの説明

データ名

コメントを追加するデータの指定

n　：　追加するコメントの文字数を指定。

m　：　領域Aからの開始位置。

使用例

```
    ...
  CAREA  80, A(1)
  ADD  TEXT
  ¥ END
  CWA     S01        :  データS01へコメントを追加する。
    ...
```

3.2.29 COMMDコマンド

記述形式

COMMD　　データ名

〔CD〕

機能

コメントを消去する。

オペランドの説明

データ名

このデータのコメントが消去される。

使用例

```
    ...
  COMMD   S01
  CD      S01・▼ABC▼
    ...
```

3.2.30 グラフィック・データ設定コマンド

記述形式

GRAPHX　　データ名〔／変数名〔，初期値，終値，増分値〔，T〕〕〕

〔GX〕

GRAPHY　　データ名
〔GY〕

GRAPHZ　　〃
〔GZ〕

機能

図形処理データの設定を行う。また作図時のタイトルの設定も可能である。

コマンドの説明

GRAPHX　：　xデータの設定
〔GX〕

GRAPHY　：　yデータの設定
〔GY〕

GRAPHZ　：　zデータの設定
〔GZ〕

オペランドの説明

データ名

図形処理データの存在するデータを指定する。

変数名

変数名に対応するデータの指定。

初期値，終値，増分値

データ設定時の制御変数。

T

データの持つコメントを図のタイトルに指定。

／以下のオペランドが省略された場合，データの持つ全データが設定される。

使用上の注意

2次元図形処理時には，xデータ，yデータそれぞれ1300点以内であること。また3次元図形処理時にはxデータ，yデータは50点以内であり，zデータは2500点であること。

使用例

データ名がS02でその入出力形式が

((ZZ(I,J),I=1,15),J=1,10),(XX(K),K=1,15),(L,L=1,10)

である場合，xデータにXX，yデータにL，zデータにZZを設定し3次元図形を描く。

GX　S02／XX

GY　S02／L

GZ　S02／ZZ

G　OPEN

G　3D

G　CLOSE

3.2.31 グラフィック制御コマンド

記述形式

GRAPH
〔G〕

{ OPEN
O
2D
省略
ADD
A
ADDA
3D
ERASE
E
CLOSE
C }

機能

画像処理の制御を行う。

オペランドの説明

OPEN 〔O〕	グラフィック処理の開設を行う。
2D	2次元図形を描く，但しxデータとyデータの設定がなされていること。
ADD 〔A〕	現在描かれている図に重ねて，2次元図形を描く。
ADDA	ADDと同様の処理を行い，y軸を追加して描く。
ERASE 〔E〕	図形消去を行い，グラフィック・パラメータをリセットする。
3D	3次元図形を描く，但しxデータとyデータとzデータの設定がなされている事。
CLOSE 〔C〕	グラフィック処理を終了する。
省略	2Dの処理をする。

使用例

データ名がS01でその入出力形式が（XX(I)，YY(I)，I＝1，100）である場合。

```
...
GX   S01/XX    :  xデータを設定する。
GY   S02/YY    :  yデータを設定する。
G    OPEN      :  グラフィック・オープン
G    2D        :  描く
...
```

3.2.32 グラフィック・パラメータ設定コマンド

記述形式

GRAPHSET
〔GS〕

- LBLX
- LX
- LBLY
- LY
- LBLZ
- LZ
- TITLE
- TL
- LIMS
- LIM
- SIZE
- SIZ
- LOG
- 3D

機能

作図時の各種のパラメータの読み込みを指定する。

オペランドの説明

LBLX
LX　：　z軸に描くラベルの指定。

LBLY
LY　：　y軸に描くラベルの指定。

LBLZ
LZ　：　z軸に描くラベルの指定。

TITLE
TL　：　図のタイトル指定。

LIMS
LIM　：　xmin, xmax, ymin, ymax。

SIZE
SIZ　：　x軸，y軸の分割数。

LOG　：　z軸，y軸のスケール，0は線型，1は常用対数軸。

3D　：　zmin, zmax, theta, gamma, Q, QS

使用上の注意

ラベル，タイトル情報はA4フォーマットで読み込まれる。数値情報はフリーフォーマットで読まれる。

GRAPHSET コマンドによる，各種のパラメータ設定が無い場合，またはGRAPH ERASE（消去）コマンドが発せられた場合，以下の値を持つ。

x軸ラベル　：　ブランク

y軸ラベル　：　ブランク

z軸ラベル　：　ブランク

タイトル　：　ブランク

xmin xmax　：　自動決定

y min　y max	:	自動決定
x 軸分割数	:	10
y 軸分割数	:	7
x スケール	:	0（線型）
y スケール	:	0（線型）
z min　z max	:	自動決定
theta　gamma	:	45°　45°
Q　QS	:	3.0　3.0

使用例

GS　LBLX

ラベルを打ち込む

GS　LOG

O　1

4. おわりに

本報告はとりあえず現状のデータプールについてまとめたものである。この種のプログラムは利用者の使用経験を十分に取入れたものでなければうまくは使えない。データプールも現状からまだまだ変化してゆくものと思われる。関係者諸氏の御意見をうかがって今後さらに改良してゆきたい。

謝　　辞

データプールの開発は東海研究所原子力コード委員会総合化専門部会の作業としておこなわれたものである。

データプールの仕様設計と開発作業の進捗は桂木学専門部会長（現安全解析部長）をはじめ総合化専門部会の専門委員である以下の方々に負うところが多い。

部会名	氏名	機関名
原子力コード総合化専門部会	○桂木　学	原研安全解析部
〃	下桶敬則	〃 原子炉データ解析室
〃	鴻坂厚夫	〃 安全性コード開発室
〃	荒井長利	〃 多目的炉設計研究室
〃	常松俊秀	〃 理論解析研究室
〃	浅井　清	〃 計算センター
〃	竹田辰興	〃 理論解析研究室
〃	小山謹二	〃 高速炉物理研究室
〃	大西信秋	〃 反応度安全研究室
〃	鈴木友雄	〃 原子炉システム研究室
〃	宮坂駿一	〃 原子炉工学部

またデータプール使用上の問題点と改善方法については原子炉工学部炉システム研究室長石黒幸男氏，同炉物理研究室土橋敬一郎氏に負うところが多い。とくにデータプールが現在の水準に到達したのは土橋氏の努力と助言による。

また計算センターの平川隆室長には開発作業を進めるうえで大変お世話になった。

以上の方々に深く感謝致します。

4. おわりに

本報告はとりあえず現状のデータプールについてまとめたものである。この種のプログラムは利用者の使用経験を十分に取入れたものでなければうまくは使えない。データプールも現状からまだまだ変化してゆくものと思われる。関係者諸氏の御意見をうかがって今後さらに改良してゆきたい。

謝辞

データプールの開発は東海研究所原子力コード委員会総合化専門部会の作業としておこなわれたものである。

データプールの仕様設計と開発作業の進捗は桂木学専門部会長（現安全解析部長）をはじめ総合化専門部会の専門委員である以下の方々に負うところが多い。

部会名	氏名	機関名
原子力コード総合化専門部会	○桂木　学	原研安全解析部
〃	下桶敬則	〃 原子炉データ解析室
〃	鴻坂厚夫	〃 安全性コード開発室
〃	荒井長利	〃 多目的炉設計研究室
〃	常松俊秀	〃 理論解析研究室
〃	浅井　清	〃 計算センター
〃	竹田辰興	〃 理論解析研究室
〃	小山謹二	〃 高速炉物理研究室
〃	大西信秋	〃 反応度安全研究室
〃	鈴木友雄	〃 原子炉システム研究室
〃	宮坂駿一	〃 原子炉工学部

またデータプール使用上の問題点と改善方法については原子炉工学部炉システム研究室長石黒幸男氏，同炉物理研究室土橋敬一郎氏に負うところが多い。とくにデータプールが現在の水準に到達したのは土橋氏の努力と助言による。

また計算センターの平川隆室長には開発作業を進めるうえで大変お世話になった。

以上の方々に深く感謝致します。

付録A　コマンド一覧

1. バッチ処理関係

区分	FORTAN拡張文名	パラメータ	機能概要
定義文	FD	Fnn＝データブール名，LRECL＝レコードサイズ／{L/E/U}，MAXRCD＝最大レコード数〔，MODE＝{▼OLD▼/▼NEW▼}，OWNER＝所有者名，DIRECT＝ディレクトリの個数〔，MACHINE＝計算機名／{▼WORD▼/▼BYTE▼}〕〔，DEVICE＝装置名／レコードギャップ／1トラック当りのバイト数〕〕	使用するデータブールの属性を定義する。
入出力文	WRITE	（{データ名/*/#}〔，P＝優先権〕〔，EXT＝展開形式〕〔，FMT＝書式名〕〔，ERR＝文番号〕）　出力並び	データブールにデータを格納する。
	READ	（{データ名/*/#}〔，ERR＝文番号〕）　入力並び	データブールからデータを読む。
	COMMW	（{データ名/*/#}〔，P＝優先権〕〔，ERR＝文番号〕）　{▼文字列▼/n，変数名}	注釈文を格納する。
	COMMR	（{データ名/*/#}〔，ERR＝文番号〕）n，変数名	格納されている注釈文を読む。
	SEARCH	（データ名，データ長〔，ERR＝文番号〕）	指定されたノード名をさがす。
ノード位置設定文	POINT	（データ名，D_{max}，D，v，rc）	ノード名探索の開始点を指定する。
	NEXTH		ノード位置設定ポインタを水平方向右側に移動する。
	NEXTV		ノード位置設定ポインタを下方向左端に移動する。
	PRIORH		ノード位置設定ポインタを水平方向左側に移動する。
	PRIORV		ノード位置設定ポインタを上のノードに移動する。

区分	FORTRAN拡張文名	パラメータ	機能概要	頁
ノード位置設定文	CONNECT	(データ名1, データ名2, RC)	二つのデータ名を結びつける。	
	DISCON	(データ名1, データ名2, RC)	CONNECT文で作られた結びつきを解く。	
	RNEXT		ノード位置設定ポインタを現在の位置からCONNECT文で結びつけられているノードに移す。	
	FIND		ノード位置設定ポインタが示しているノードのノード名を知る。	
	DNEXT	(変数名)	一次元インデックス位置設定ポインタを現在の位置から次の位置へ移す。	
保守管理文	COMDENSE	(データブール名, WF=nn)	データブール内の各所に散在する使用不可領域を集め，使用可能領域とする。	
	SORT	(ノード名〔, LEVEL={i / ALL}〕*, WF=nn〕〔, SR=m〕)	指定されたレベルのノード名を並べかえる。	
	RECOVER	(データブール名〔, TYPE={W / S, WF=nn / C, WF=nn}〕)	使用不可となったデータブールを再使用可能にする。	
宣言文	OPTION	〔BUFFER=作業用領域サイズ/{L/E/U}, 〕〔LRUSIZE={30 / LRUテーブルサイズ}, 〕 〔MAXUDP=最大使用データブール数, 〕〔QMAIN={CREATE / NOCRATE}〕	データブール・アクセス・ルーチンの動作環境を設定する。	
	SYN	v1=データ名1〔, v2=データ名2, ……〕		
	ALIAS	v1=予約語1〔, v2=予約語2〕	予約語を変更する。	

2. ＴＳＳ処理関係

コマンド名 〔コマンド名省略形〕	オペランド	説明	ページ
STOP	〔▼メッセージ▼〕	ＤＰＳ実行終了	
PROCEDURE 〔PROC〕	プログラム	プログラム定義文	
MANUAL	〔▼メッセージ▼〕	Manual モードへ戻る	
GOTO	文番号	ＧＯＴＯ文	
END		プログラム定義の終り	
RUN	プログラム名〔▼L▼〕	プログラム実行文	
ATTACH 〔A〕	ファイル定義名 {,/=} データプール名 , LRECL＝nnn〔／{L/U}〕, MAXRCD＝mmm	データプールの割り付け	
ALLOCATE 〔ALLOC〕	ファイル定義名 {,/=} データプール名 , LRECL＝nnn〔／{L/U}〕, MAXRCD＝mmm , MAXDT＝kkk	データプールの作成	
SYNONYM	変数名＝ノード名〔．ノード名〔．～〕〕	同義語定義文	
変数名＝	▼文字例▼ {{変数名/定数}〔{+/−/*/／}{変数名/定数}〕}	代入文	
CATLIST 〔CATL〕	データプール名	データプール情報の印刷	
LIST	データ名〔／{C, F, P, pRn}〕	データ情報の印刷	
LISTD 〔LD〕	データ名〔／変数名 〔，初期値，終値，増分値〕〕	数値データの印刷	
COMPARE	データ名，データ名	比較	
MOVE	データ名，データ名	移動	
COPY	データ名，データ名	複写	
DELETE 〔DELE〕	データ名	消去	

コマンド名 〔コマンド名省略形〕	オペランド	説明	ページ
REAME 〔R〕	データ名，データ名	改称	
CHAP	データ名〔／{ALL / A} / {ONLY / O}〕，{m / +m / −m}〔，pRn〕	プライオリティの変更	
ON	ERROR {GOTO 文 / STOP 文 / MANUAL文}	エラー処置	
ON	{EQUAL / UNEQUAL} {GOTO 文 / STOP 文 / MANUAL文}	比較後の処置	
ON	({変数名 / 定数}・Re・{変数名 / 定数}) {GOTO 文 / STOP 文 / MANUAL文}	条件文	
COMMAREA 〔CAREA〕	n，A〔(m)〕	テキストの読み込み	
テキスト		コメント・テキスト	
¥END		テキスト終了文	
DISPLAY 〔DISP〕	〔n，A〔(m)〕〕	テキスト表示	
CEDIT	▼▼文字列▼▼文字列▼▼	文字列の置換	
COMMR 〔CR〕	データ名，変数名〔，A(m)〕	コメントの読み込み	
COMMW 〔CW〕	データ名，n〔，A(m)〕	コメントの書き込み	
COMMWA 〔CWA〕	データ名，n〔，A(m)〕	コメントの追加書込	
COMMD 〔CD〕	データ名	コメントの消去	
GRAPHX 〔GX〕	データ名〔／変数名 〔，初期値，終値，増分値〔，T〕〕〕	xデータの読み込み	
GRAPHY 〔GY〕	同　上	yデータの読み込み	
GRAPHZ 〔GZ〕	同　上	zデータの読み込み	

コマンド名〔コマンド名の省略形〕	オペランド	説明	ページ
GRAPH〔G〕	OPEN〔O〕	グラフィック・オープン	
同上	2D〔省略〕	2次元図形のプロット	
同上	ADD〔A〕	重ね図	
同上	ADDA	重ね図，Y軸を追加	
同上	3D	3次元図形のプロット	
同上	ERASE〔E〕	画面清去，パラメータ・リセット	
同上	CLOSE〔C〕	グラフィック・クローズ	
GRAPHSET〔GS〕	LBLX〔LX〕	x軸ラベルの読み込み	
同上	LBLY〔LY〕	y軸ラベルの読み込み	
同上	LBLZ〔LZ〕	z軸ラベルの読み込み	
同上	TITLE〔TL〕	図のタイトル読み込み	
同上	LIMS〔LIM〕	x min，x max，y min，y max を読む	
同上	SIZE〔SIZ〕	x軸，y軸の分割数読み込み	
同上	LOG	軸スケールの読み込み	
同上	3D	3次元パラメータの読み込み	

付録B　情報検索システム作成例

データプールの情報検索システム用ファイルの作成例と使用例を示す。キーワードは2つのグループに分けることができる。

(1) CPC（Computer Physics Communications）に記載されている論文，数学的サブルーチンを検索。

(2) 対象としているCPCは第1～11巻までで，現在は第17巻まで発行されている。

(3) 検索はキーワード（物理キーワードと数学キーワードの2グループ）を手がかりにおこなう。

(4) キーワード一覧表を自動的に作成する。

(5) 検索の対象となるデータはデータプールと呼ばれる構造化されたファイルにおさめられている。

ターミナルからの使用法

プログラムは実行中を通じてKEYモードかLISTモードにあり，各々についてPhysics，Mathematicsの2つの状態がある。どの状態にあるかはターミナルから打ち出される文字によって判断できる。

KEY (P)	:	KEYモード	Physics 状態
KEY (M)	:	〃	Mathematics 状態
LIST(P)	:	LISTモード	Physics 状態
LIST(M)	:	〃	Mathematics 状態

KEYモードかLISTモードかにより入力できるものは異なる。Physics，Mathematicsいずれの状態でもキーワード（又はその連番）が異なるだけで形式は同じなので一緒に記述する。

1. KEY（　）モードのとき，コマンド又は論理式を入力する。

①	@FIN	:	このセッションを終了させる。
②	@P	:	Physics 状態に変える（既にPhysics 状態ならば意味をもたない）
③	@M	:	Mathematics 〃 （〃 Mathematics 〃 〃 ）
④	論理式	:	上記①～③のいずれでもないときは論理式とみなす。 論理式はキーワード（又はその連番）と演算子＊，＋，（，）を用いて記述する。勿論キーワードはその時の状態（Physics かMathematics）に対応するものを用いること。

2. ＬＩＳＴ（　）モードのとき，サブコマンド又はカタログ名（又はサブルーチン名）を入力する。

① ＃ＥＮＤ　：　このＬＩＳＴモードを終了してＫＥＹモードに戻る。

② ＃ＡＬＬ　：　条件を満足するすべてのカタログ（又はサブルーチン）をターミナルに出力する。

③ カタログ名：（又はサブルーチン名）　上記２つのサブコマンド以外の時はカタログ名（又はサブルーチン名）とみなされる。これは論理式の結果として出力されたもののうちの１つでなければならない。Mathematicsの場合同名のサブルーチンが複数個あるときは，その後にカッコでくくってカタログ名を与える。与えないとその名前のルーチンのすべてが出力される。

使用例

```
$$CPS J1666
+ PASSWORD ?=■■■■■■■■■■■
CPS JOB NO.=DTK03.0348
. CPS(V-10/L-04) INITIALIZATION NORMAL END
. READY*FOR OUTPUT:NONE
*.CPCKEY   ←―― マクロ・コマンドの起動である

 PLEASE WAIT 15 TO 30 SECONDS FOR INITIAL LOADING.
             3            4            20            80
MEMORY (USED/RESERVED)=( 19935/ 25000).
KEY(P)   ←―― 物理キーワードを検索できる状態
>3*4     ←―― キーワード3 AND (*) キーワード4を検索
THERE IS NO CATALOG/SUBROUTINE IN THIS CONDITION
KEY(P)
>3+4   ←―― キーワード3 OR キーワード4を検索
 AAAC  ABUE  ←―― カタログ AAAC と ABUE が存在する
LIST(P)      ←―― 出力モードに切り換った
>AAAC
**V:07,1974,200-206            C:AAAC,IBM360,433,EBCDIC                    KEY21060
T:INVERSION OF ABEL'S INEGRAL EQUATION-APPLICATION TO PLASMA               KEY21070
  SPECTROSCOPY                                                             KEY21080
A:C.FLEURIER,J.CHAPELLE                                                    KEY21090
K:PLASMA SPECTROSCOPY,ABEL INTEGRAL EQUATION,LEAST-SQUARES                 KEY21100
  APPROXIMATION                                                            KEY21110
LIST(P)    ←―― まだ出力モードである
>CHARGE    ←―― 出力モードでキーワード入力はできない
CHARGE  NOT FOUND
LIST(P)
>@M      ←―― 出力モードでモードの変更はできない
@M      NOT FOUND
LIST(P)
>*END    ←―― 出力モードを終了させる
KEY(P)
>@M      ←―― 物理キーワード検索モードから数学キーワード検索モードへ切り換える
KEY(M)   ←―― 数学キーワード検索モード
>PADE    ←―― キーワード入力
```

```
PADE
 CANTAT(ACWG)  PADE  (ACWV)  SOD   (ACRT)
LIST(M)
>SOD
**V:09,1975,46-50              C:ACRT,ICL1430,299,EBCDIC      L=AL        KEY29530
T:SUBROUTINE AND PROCEDURE FOR THE RAPID CALCULATION OF SIMPLE OFF        KEY29540
  DIAGONAL RATIONAL APPROXIMANTS                                          KEY29550
A:P.R.G.MORRIS,D.E.ROBERTS                                                KEY29560
K:GENERAL,RATIONAL,APPROXIMANT,CHISHOLM,SIMPLE OFF-DIAGONAL,              KEY29570
  PERTURBATION SERIES, PADE                                               KEY29580
*SOD                                                                      KEY29590
K:RATIONAL,APPROXIMANT,CHISHOLM,OFF-DIAGONAL,PRONG,PERTURBATION,SIERIES,KEY29600
  PADE                                                                    KEY29610
AC:THESE APPROXIMANTS MAY BE APPLIED FOR RESUMMATION OF THE               KEY29620
   PERTURBATION SERIES WITH TWO INTERACTION HAMILTONIANS,E.G.,THAT IN     KEY29630
   POTENTIAL SCATTERING GIVEN BY THE SUM OF AN ATTRACTIVE AND A           KEY29640
   REPULSIVE YUKAWA POTENTIAL; MANY OTHER APPLICATIONS ARE ENVISAGED,     KEY29650
   ESPECIALLY IN THE FIELD OF CRITICAL PHENOMENA.                         KEY29660
R:J.S.R.CHISHOLM, MATH.COMP. 27 (1973) 841.                               KEY29670
R:R.HUGHES JONES AND G.J.MAKINSON, J.INST.MATHS.APPLICS. 13 (1974)        KEY29680
  299.                                                                    KEY29690
R:P.R.GRAVES-MORRIS,R.HUGHES JONES AND G.J.MAKINSON,                      KEY29700
  J.INST.MATHS.APPLICS. 13 (1974) 311.                                    KEY29710
R:P.R.GRAVES-MORRIS AND R.HUGHES JONES, BROOKHAVEN PREPRINT (1974) AMD    KEY29720
  674.                                                                    KEY29730
R:G.A.BAKER,J.MATH.ANAL.APPLICS. 43 (1973) 498.                           KEY29740
LIST(M)
>APPROXIMANT
NAME LENGTH IS TOO LONG. SHORTEN TO 8 CHARACTERS
   A   P   P   R   O   X   I   M   A   N   T
NAME LENGTH IS TOO LONG. SHORTEN TO 8 CHARACTERS
   A   P   P   R   O   X   I   M   A   N   T
APPROXIMNOT FOUND
LIST(M)
>$END
KEY(M)
>33
```

```
33
 BSSL   (AACF)
LIST(M)
>#ALL   ←――該当する情報をすべて出力させる
**V:03,1972,173-179          C:AACF,CD64,625,CD                          KEY09710
T:THE CALCULATION OF ABSORPTION AND ELASTIC CROSS SECTIONS USING THE     KEY09720
  OPTICAL POTENTIAL                                                      KEY09730
A:A.C.ALLISON                                                            KEY09740
K:ATOMIC,POTENTIAL,COMPLEX POTENTIAL,OPTICAL MODEL,OPTICAL POTENTIAL,    KEY09750
   SCHROEDINGER EQUATION,NUMEROV,SCATTERING,ABSORPTION,CROSS SECTION,    KEY09760
  PHASE SHIFT                                                            KEY09770
*BSSL                                                                    KEY09880
K:SPHERICAL,BEESEL,NEUMANN                                               KEY09890
AC:A SUBROUTINE THAT RETURNS THE VALUE OF THE SPHERICAL BESSEL AND       KEY09900
   NEUMANN FUNCTIONS AT THE MATCHING POINTS.                             KEY09910
KEY(M)  ←――#ALL コマンド実行後は自動的にキーワード検索モードになる
>@FIN   ←――キーワード検索モードにおける この入力は システムの終了を意味する
** END OF SESSION. CPU TIME=    0.0 (SEC) **


   STOP STATEMENT EXECUTED.    777
       USED DOMAIN SIZE    1 KW
. PROGRAM NAME: NONAME   , RETURN CODE: 000.
#
```

出力データのカード形式

(1) ＊＊巻，年，ページ，カタログ番号，コンピュータ種類，カード枚数，カードコード

例

＊＊ V：01，1969，25−30，C：ABOC，CD66，547，CD

(2) タイトル

例

T：COULOMB FUNCTINS FOR COMPLEX ENERGIES

(3) 著　者

例

A：T. TAMURA，F. RYBICKI

(4) キーワード

例

K：NUCLEAR，SCHROEDINGER EQN.，REACTIONS，SCATTERING，WAVE FUNCTION，GAMOW FUNCTION，COULOMB，COMPLEX COULOMB，POTENTIAL，COMPLEX GAMMA FUNCTION.

(5) ＊サブルーチン名

例

＊COULOM

(6) キーワード

例

K：COULOM，COMPLEX，IRREGUAR

(7) 参考文献

例

R：B. BUCK，R. N. MADDISON AND P. E. HODGSON，PHIL. MAG. 5（1960）118.

(8) ＊サブルーチン名

例

＊DEFEQA

(9) キーワード

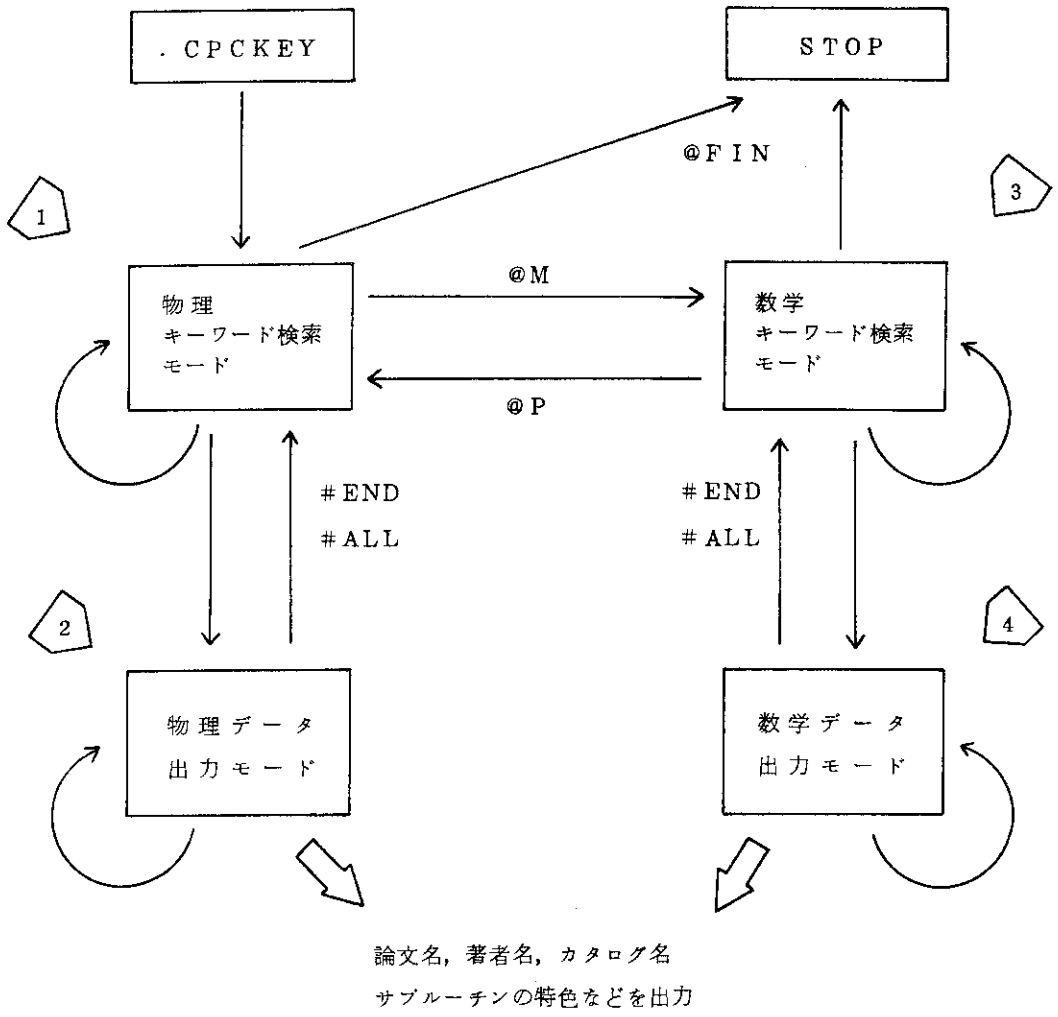
例

K：INTEGRAL，FIVE POINTS，STOERMER

(10) 参考文献

例

R：P. HENRICI，DISCRETE VARIABLE METHODS IN ORDINARY DIFFERENTIAL EQUATIONS（JOHN WILEY AND SONS，NEW YORK，1962）CHAPTER6.



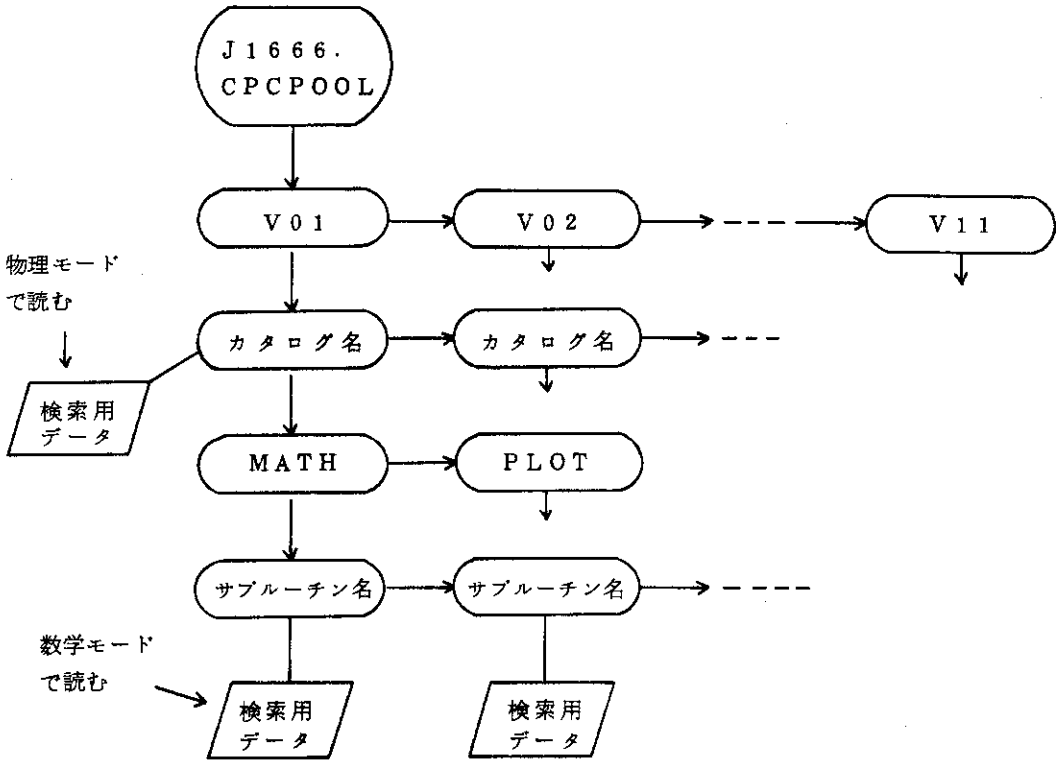
1 物理キーワード入力

2 物理データ出力

3 数学キーワード入力

4 数学データ出力

Fig. B−1 State Transition of IR System



記号 ◯ は木構造ファイルのノードを示す。

Vnn は雑誌ＣＰＣの巻番号を示す。

Fig. B−2 File Structure for Retrieval