

40 - 3
電 子 計 算
技 術 管 理

土 木 研 究 所 資 料

第 122 号

電 算 速 報

第 3 号

昭 和 40 年 10 月

建 設 省 木 土 研 究 所

目 次

§ 1. 電算運営資料	1
〔1〕 インジケータの種類と説明	1
〔2〕 ALGOL-H(H ₁)から NEW ALGOL-H(H ₂)への変換について	4
〔3〕 NEW ALGOL-Hの文法概要	12
〔4〕 FORTRAN(LV)と NEW ALGOL-Hとの比較	42
§ 2. プログラム紹介	61
〔1〕 標準手続および一般手続プログラム一覧表 およびその内容について(その2)(10月30日現在)	61
〔2〕 プログラム紹介	78
(1) 一元配置の分散分析を行なう手続	78
(2) F分布表片側5%の近似値を求める手続	81
(3) Dean Marston 公式による土圧の算定(project形)	84
(4) パワースペクトル計算	88
§ 3. 電算室よりのお知らせ	93
〔1〕 プログラマー登録番号追加一覧表	93
〔2〕 標準手続追加一覧表	96

§1 電算運営資料

(1) インジケータの種類と説明

インジケータは、本体関係のもとと入出力チャネル関係のもととに大別される。ある状態が発生するとこれらのインジケータが「ON」(1の状態)になる。

また、インジケータが「ON」になつても、割込みの原因とならないものもある。ここでは、それらについて、表にまとめてかかげた。

① 本 体 関 係

本体関係のインジケータは、SIR(Set Indicator Register)にまとめられており、SIRの各ビットがひとつひとつのインジケータとなっている。これをSI(Set Indicator)という。SIRは、42ビットである。

a. 割込みの原因となるもの

SIの種類	原 因
MMパリティ・エラー	メイン・メモリのパリティ・エラー
AM "	ARのパリティ・エラー
SM "	B/D変換のときのパリティ・エラー
無効の命令	命令コード以外のコードがきた場合
無効のアドレス	収容メモリー以上の番地指定がなされた場合、および ロックアウト領域に書き込もうとした場合
A-OFW	固定小数点演算の結果、ARオーバーフローを生じた場合。 ただし、除算、およびASP, SSP, ALI, SLI命令を除く。
Q-OFW	固定小数点形の除算命令で $ (AR \sim AR+1) \geq (n) $ であつた場合
ゼロ・ディバイザー	除算命令において $(n) = 0$ の場合
E-OFW	浮動小数点演算において、指数部がオーバーフローした場合
E-UFW	浮動小数点演算において、指数部がアンダーフローした場合
F-OFW	○ASP, ALI 命令(符号をもたない演算)の実行の結果、オーバーフローが生じた場合

電算運営資料

SIの種類	原因
F-UPW	・非正規形 (unnormalized) の浮動小数点演算において、仮数部オーバーフローした場合 ・SSP, SLIの命令において、$(AR) < (n)$ であつた場合。(符号をもたない演算のため) ・非正規形の浮動小数点演算において、仮数部がアンダーフローした場合
マニュアル インタラプト (1)	マニュアル インタラプト スイッチ(1)を押した場合 } 同じく (2)~(5)を押した場合
" (2)	
" (3)	
" (5)	
リアル・タイム・クロック (RTC)	リアル・タイム・クロック・カウンタが0になつた場合
デイ・タイム・クロック	時計が $5 \times n$ 分 ($n = 0, 1, 2, \dots$) をさしたとき

b. 割込みの原因とならないもの

SIの種類	原因
プログラム スイッチ(1)	・プログラム スイッチ(1)~00を押した場合 ・マニュアル スイッチ(1)~00を押した場合
" 00	
マニュアル スイッチ(1)	
" 00	
桁上り (PC)	ABC, SBC, ESの各命令の実行の結果、それぞれの現象が生じたとき
桁借り (PB)	
サイン (PS)	

(注) 上記の外に、SSIJ命令により、任意のSIを「1」にすることができる。ただし、マニュアル スイッチに関しては無効である。

② 入出力チャネル関係

入出力チャネルに関するインジケータは、各チャネルごとに161~1614をもっている。ただし、チャネルにより、設けられていないインジケータもある。同じ番号のインジケータは、原則として同種のエラー、または状態を表わしている。チャネルごとに設けられているインジケータの種類に

については、下表を参照されたい。なお、MASHにおいてI/Oマ
クロ命令を使用するときは、考慮する必要はない。

a. 割り込みの原因となるもの

インジケータの種類	番号	原因
I/O END	1	そのチャネルに関する命令が終了したとき。 ただし、命令語のT (TAG) に“ I ”を指定し た場合のみ
I/O パリティエラー(1)	2	チャネル動作の際、本体のメモリーにパリテ イ・エラーを生じた場合
I/O パリティエラー(2)	3	チャネル内のデータ移動中にパリティ・エラ ーを検出した場合
無効の I/O 命令	4	○ I/O 命令コードでないコードがきた場合 ○ チャネル番号の指定ミス ○ ユニット番号が無意味な場合
無効の I/O アドレス	5	○ 入出力命令またはその補助語に指定したアド レスが、収容メモリー以上であつた場合 ○ メモリー・ロック・アウトされた領域に書き 込もうとした場合
アラーム	6	ユニットがアラームの状態にあつた場合、すな わち、電源「OFF」、リボンエンド、ペーパーエンド など
EOF	8	磁気テープチャネルにおいて、MTBT, MTFT MTRL, MTRU 以外の命令の実行中にエンド・ オブ・ファイル (E. O. F) マークを検出したとき。
EOT	9*	テープエンド箱を検出したとき
BOT	10	ビギニング箱を検出したとき。ただし、Rewind 命令を除く
エンブテイ・ホツパー	11	カードリーダー、カードパンチにおいて、ホツパーにカー ドがなくなつたとき
フル・スタツカー	12	カードリーダー、カードパンチにおいて、スタツカーが一 ぱいになつたとき
ミス・フィード	13	カードリーダー、カードパンチにおいて、何かの原因 でカードがフィードミスを起こしたとき
マスター・カード	14	カードリーダー・チャネル内で、マスターカード用のスイ ッチが「ON」になつているときに、マスターカード を読み込んだ場合。(マスター・カードについ

電算運営資料

インジケータの種類	番号	原 因
		ては、INSTRUCTION MANUALに述べられている)

* 注 ドラム・チャネルにおいては、ドラム・アドレス・インバリッドが検出されると9がセットされる。

b. 割り込みの原因とならないもの

インジケータの種類	番号	原 因
ビ ジ ー	7	チャネル用の固定番地 $n_1 \sim n_2$ (INSTRUCTION MANUAL参照) に命令が残っており、チャネルが、次の命令を受け入れることができない場合

[2] ALGOL-H(H_1) から NEW ALGOL-H(H_2)
への変換について

中 村 肇 一

は じ め に

本年12月よりNEW ALGOL-Hが使用できる見込みである。現在のALGOL-Hを包含し、更に大巾に文法制限が解除されているので非常に使いやすいものになっているが、以下ALGOL-Hとの対比を中心としてその文法の概要をのべる。

1. 記号類

- 1.1 論理値はALGOL-60, ALGOL-Hの true, false に対し TRUE, FALSE となつた。
- 1.2 論理演算子はALGOL-H に対し $\equiv$ が追加されALGOL-60 並みとなつた。
- 1.3 区切子にwhileが追加されALGOL-60 並みとなつた。

- 1.4 宣言子はALGOL-H同様 own が除かれている。
- 1.5 指定子に string, label が追加されALGOL-60 並みとなつた。
- 1.6 その他 code, external, attach が追加された。これらは
ALGOL-60にないもので codeは機械語との接続, externalは内蔵された標準プロセデュアの冒頭宣言に、attachは新たに機械語に訳したプロセデュアを標準プロセデュアとして内蔵するための冒頭宣言に用いる。
2. 変 数
配列の添字に書かれた式はその結果が real となるときは自動的にFIXして integer とする。
3. 関数指示子
 - 3.1 新たに追加されALGOL-60 並みとなつた。手続宣言とほとんど同じであるが一箇関数の場合答を受ける名前をとつて、手続本体の中で手続自身の名前に結果を代入する様にして、使用に当つては式の中で用いる。宣言は procedure の前に< type >をつける。
 - 3.2 宣言のとき) < letter string > : (をパラメタ部に書くことは禁止する。
4. 標準関数
ALGOL-H と同じく大文字を用いるが、ALGOL-60 同様 real integer は何でもあつても答はすべて real (SIGNのみ integer) となる。
5. 換算関数
ALGOL-H 同様 FLOAT, FIXを用いる。
6. 式
 - 6.1 < if clause >に if BE then の他に on < unsigned

電算運営資料

integer > then が追加された。これは ALGOL-60 にもないものでマニュアルスイッチの 1~10 の何れかが押してあればそのときには then 以下を実行するものである。

- 6.2 ÷ の両辺は integer に限り結果も integer となるが他の / + - * の両辺は integer real 何れでもよく結果は / 以外の両辺が integer のときは integer で他はすべて real となり ALGOL-60 並みとなつた。

但し条件式では else の前後の型は同じでなければならない。

- 6.3 関係演算子の左右の型が異るときは real として比較される。

7. 代入命令

多重代入命令の左辺の型は同じでなければならないが、左辺と右辺の型が同じでないときは自動的に左辺に合わせて型変換を行ない、ALGOL-60 並みとなつた。

8. GOTO 命令

スイッチ指示子で値が real となつたときは自動的に FIX し、未定義のときは誤りとする。

9. 条件付命令

< if clause > に on < unsigned integer > then が追加された。条件式の場合と同じである。

10. FOR 命令

10.1 step until エレメントの中の算術式に何をを用いてもよいことになつた。又型は何であつてもよい。

10.2 while エレメントが追加され

for I := AE while BE do S;

が可能となつた。

10.3 コントロール変数が添字付変数でも良いことになつた。

11. 手続命令

11.1 パラメタに string, label が追加された。string は対応する実パラメタに * <letter string> * を書くところの字が印刷に使用され、label に対応する実パラメタによつて行先を指定できる。

11.2 パラメタとしての procedure はパラメタを持つもので良いことになった (ALGOL-H ではパラメタなしのものに限つた)。

11.3 Call by value で array を呼べないことは H1 同様であるが、call by name のパラメタに対応する実パラメタに式を許すことになった。

但し対応するパラメタの型、種類は同じでなければならない。

12. スイッチ宣言

スイッチリストは指示式を認めることになり ALGOL-60 並みとなつた (H1 ではレーベルに限つた)

13. 手続宣言

13.1 指定子として string, label が追加された。

13.2 機械語との接続を可能にするため手続宣言の中で code <octal>; という命令が追加された (ALGOL-60 にない)

octal は 0~7 の数字を 14 個またはその倍数並べたもので各 42 ビットの 1 語に対応する。

13.3 手続本体で global な量を使用するときはすでに宣言されたものでなければならない (H1 と同じ)

13.4 external procedure 宣言の body では global な量を使用してはならない。

13.5 external 宣言はあらかじめ翻訳しておいた procedure を使用するための宣言で

external 手続宣言 external I, I, ...;

external 関数指示子宣言 external <type> I, I, ...;

で、このように名前を宣言してから使用すれば計算機はあらかじめ

電算運営資料

M T (磁気テープ) に格納してある手続類を引き出してきて計算に使用することになる。

13.6 external 用の格納

今後一般的に使用するために M T に格納するためには

```
attach <procedure declaration> end;
```

とすれば翻訳結果はシステムテープに貯えられ、後に external 宣言があれば使用可能の状態になる。

attach は begin 扱いなので必ずこれに対応する end が必要である。

14. I/O 関係

14.1 読み込み

数値 READ (A, B, ...);

integer real いづれでも変数の型に合わせてデータテープの値を変換格納する。

ブリーヤンの読み込みは F や T が先頭にあればよい。

文字 アルファモードの読み込みは 7 文字を単位としてデータテープ上

= K I N $\bar{O}$, = T A N I # A, ...

等と=の後に 7 文字までを入れば指定された変数 (integer タイプ) に読み込み可能である。

文字数字何れでもよく文字間のスペースは # でやはり一文字である。

7 文字全部ブランクにするには =, とすればよい。

14.2 印刷 (一般)

3 ケタの integer, 3.2 ケタの real I, A に対しては

```
PRINT ('-3Z, -3Z, 2D', I, A);
```

とする。前はフォーマット指定部で ' ' につまれ後の変数に対応して 2 種のフォーマットがカンマで切られている。

フォーマット指定では

- (又は十何れでもよい)	符号を打て
Z	ゼロサプレスする数字
D	ゼロサプレスしない数字
B	ブランク
/	復帰改行
↑	用紙上端よりの印刷指定
F	Boolean の印刷指定
10	スモールテン指定

たとえば用紙上端より I, A を上例の様に印字し中間にスペース 2 個を入れて一行に印刷するなら

```
PRINT('↑-3Z2B,-3Z,2D/', I, A);
```

となる。

Z, D, B, /, F はその前に特号なしの整数をつけて倍数とすることができる。A が 5 個分 A 1, A 2, ... A 5 あるなら

```
PRINT('↑-3Z2B,5(-3Z,2D)/', I, A1, A2, A3,
      A4, A5);
```

となる。

Boolean は B が TRUE なら

```
PRINT('5F/', B);
```

で TRUE と、FALSE の場合は FALSE と印字され F が 1 個なら先頭の T、F だけ印字する。

14.3 印刷 (簡便法)

フォーマットが面倒なら

integer に I、real に R、Boolean に L を用い

```
PRINT('I', A); PRINT('R', B); PRINT('L',
      C);
```

でよい。

integer は □□±XXX.....X

13 ケタ (ゼロサプレス)

Boolean は □□TRUE □ 又は □□FALSE

real は $\square\square\pm.\underbrace{\times\times\dots\times}_{4\text{ケタ}}\times_{10}\underbrace{\pm\times\times}_{2\text{ケタ}}$

と印刷する I, R, L もその前に符号なしで整数を入れて倍数を示すことができる。前の例は

```
PRINT('↑I, 5R/', I, A1, A2, A3, A4, A5);
```

となる。

14.4 印刷 (文字の印刷 a)

$I = \times\times\times\square\square A = .\times\times\times\times_{10} \times\times$

の形で印刷したいときは

```
PRINT('↑I=' -3Z2B, 'A=' -4D10 -2D/', I, A);
```

文字だけを印刷したいときは

```
PRINT("ALPHA");
```

とすればよい。10 のスペースや 3 行の復帰改行は

```
PRINT('10B');
```

```
PRINT('3/');
```

でよい。

文字の印刷は 5 文字の ALPHA に対し

```
PRINT('5S', 'ALPHA');
```

としてもよい。

14.5 印刷 (文字の印刷 b)

アルファモードで入っている情報を印刷するにはたとえば C に 6 文字の情報が入っていれば

```
PRINT('6A', C);
```

でよい。7 A とすれば最後がスペースになる。7 以上は不可で、このときは 2 語に分ける。

C の内容が PQR というアルファモードのものであるかどうかによつて S に 1 を加えるなら

```
if C=EQUIV(PQR) then S:=S+1;
```

とすればよい。

途中で C に XYZ というアルファモードの情報を代入したければ

integer の C に対し

C := EQUIV (XYZ);

でよい。空白をアルファモードで代入するには EQUIV(##)とする。

14.6 印刷 (変数による制御)

変数によつて R, I, L, B, Z, D, /, A, S の倍数を制御し

たいときは

命令中で N 個スペースをおいてから I を打ち M 回復帰改行するなら

REPX(N, M);

PRINT('XB, IX/', I);

でよい。X の所には順に N と M が代入される。

14.7 磁気テープ関係

PUT(<address>, I);

GET(<address>, I);

のほかに

MTWRITE(<unit>, <format string>, <list>);

MTREAD(<unit>, <format string>, <list>);

も使用できる。

15. その他

15.1 halt; ①①は使用可

15.2 array 読み込みは READ(I); とすれば宣言された通りの大きさ
に array を読み込む。宣言を変えるのが面倒ならたとえば 3 次元で
N, L, M を読んでから A[I, J, K] に対し代入するのに
READARRAY(A, 1, N, 1, L, 1, M); でもよい。

16. ALGOL-H(H1) と NEWALGOL-H(H2) に於ける PRINT 形式 の対照

ALGOL-H(H1)	NEW-ALGOL-H(H2)
-------------	-----------------

CRLF(5);	PRINT('5/');
----------	--------------

SPACE(4);	PRINT('4B');
-----------	--------------

電算運営資料

```

ALGOL-H(H1)          NEW-ALGOL-H(H2)
PRINT(3.4,A);         PRINT('B-3Z, 4DB',A);
PRINT(4,A);           {※Aがintegerの場合
                       PRINT('B-4ZB',A);
                       ※Aがrealの場合
                       PRINT('B-. 4D10-2ZB',A);

PRINT('BANK'); PRINT("BANK");
PRINT(7,BE); PRINT('B5FB',BE);

```

[3] NEW ALGOL-H の文法概要

中 村 慶 一

はじめに

ALGOL-60 の文法書は“である”という意味を持つ::=と、“又は”という意味を持つ|と、<>記号の中に書かれた我々の言葉並びにALGOL-60 で許された単語記号等で記述された文章論とこの補足を行つた意味論から成り立っている。特に前者はこの報告者の名をとつて、Buckus notationと云われ文法の記述方法としては極めて厳密である代りに一見難解である。たとえば

$$\langle \text{identifier} \rangle ::= \langle \text{letter} \rangle | \langle \text{identifier} \rangle \langle \text{letter} \rangle | \langle \text{identifier} \rangle \langle \text{digit} \rangle$$

という記述から「<名前>とは<文字>を先頭とし、場合によつてはその後に<文字>や<数字>が続いたものである」という意味をくみ取るまでには多少の時間を要する。

NEW ALGOL-HはALGOL-60 の文法を大巾に取り入れ、部分的にはALGOL-60 がない便利な部分を付加した文法を採用しているが、その文法記述は例によつてBuckus notation で書かれているので、多少の不

厳密さをしのでこれを平文に直し、特にALGOL-60との対比されるべき点(制限された所、付加された所)について触れてみたい。

1° 基本記号(basic symbols)

基本記号としては文字、数字、論理値、限定詞の4種がある。

1.1 文字(letters)

大文字のAからZまでの26文字(ALGOL-60では小文字も含む)

1.2 数字(digits)

0から9までの10種

1.3 論理値(logical values)

TRUE及びFALSE(ALGOL-60ではtrue及びfalse)

1.4 限定詞(delimiters)

限定詞には演算子、区切り子、括弧、宣言子、指定子の5種がある。

1.4.1 演算子(operators)

演算子には算術演算子、関係演算子、論理演算子、逐次演算子の4種がある。

1.4.1.1 算術演算子(arithmetic operator)

+ - * / ÷ ↑

(ALGOL-60では掛算記号は×)

1.4.1.2 関係演算子(relational operator)

< ≤ = ≥ > ≠

1.4.1.3 論理演算子(logical operator)

≡ ⊃ ∨ ∧ ¬

1.4.1.4 逐次演算子(sequential operator)

go to if then else for do on (onはALGOL-60にない)

1.4.2 区切り子(separator)

, . 10 : ; := # step until while
comment (ALGOL-60では#の代りに␣)

電算習資料

1.4.3 括弧 (bracket)

() [] * * begin end

1.4.4 宣言子 (declarator)

Boolean integer real array switch
procedure external

(ALGOL-60 には own があり、external はない)

1.4.5 指定子 (specifier)

value label string

1.4.6 其他 code halt (何れも ALGOL-60 にはない)

1.5 保留語 (reserved word)

以下の単語は保留語として特定の意味を持つので名前として使用してはならない。

論理値 TRUE, FALSE

標準関数 SIN, COS, ARCTAN, EXP, SQRT, SIGN, LN,
ABS

(ALGOL-60 ではすべて小文字であつた)

換算関数 FIX, FLOAT; EQUIV

(ALGOL-60 では FIX の代りに entier がある)

I/O関係 READ, READARRAY, PRINT, PUT, GET,
MTREAD, MTWRITE, REPX

2° アルゴルの基本原則

1. 我々の文章、文、句に対応してアルゴルではプログラム (program)、命令 (statement)、式 (expression) がある。また単語、記号に当るものとして限定詞が、名詞に当るものには名前があり名前と限定詞をくみ合せて式が、式と限定詞をくみ合せて命令が構成される。

アルゴルプログラムは1つ以上の命令から成り立ち、これらはセミコロンで区切られる (但し else の直前ではセミコロンを取らなければならない)。

2. 命令は文章の括弧 begin と end (式のカッコは丸カッコを用いる)

でくくつて1つの命令にまとめることができる。

3. アルゴリズムプログラム自身は begin と end でくくられてなければならない。
4. 名札を除くすべての名前は、その名前が使用される範囲の一番先頭の begin に続く宣言部で宣言されてなければならない。また名札は命令の前にコロンを介して置かれるがこれは名札宣言と解される。このようにして宣言された名前は、その begin に対応する end までの間についてのみ意味を持ち、その外側では意味を持たない。このことを名前はその宣言されたブロックに関し、局所的 (local) であると云う。

勿論1段内側のブロックについては、内側で同じ名前が宣言されていない限り通用し、同じ名前が宣言されていれば内側のブロックではその方が優先する。

5. 命令は GOTO 命令で行先を変更された場合を除き原則として書かれた順に実行される。
6. すべての変数は値を代入又は読込みしてからでなければ式の中で使用してはならない。
7. 1つのデータの読込みが終れば入力装置の上では次のデータが読込まれる準備がされる。
8. 下つき添字、上つき乗べき、水平分割線による割算の表示は禁止される。従つて

$$\begin{array}{ll} A_3 & BA_{4,5} \text{ は } A[3] \quad BA[4,5] \\ A^B & B^{A+2} \text{ は } A \uparrow B \quad B \uparrow (A+2) \\ \frac{A+B}{C+D} & \text{ は } (A+B) / (C+D) \end{array}$$

と表現する。

9. 取り扱い算術定数は -3.6 , 4×10^{-3} 等の実数 (real) と整数 (integer) とに分れるが、 4×10^{-3} は $4_{10}-3$ と書かれる。
10. 変数に対する値の代入は $:=$ (コロンイコール) という記号によるが、新しい値が代入されると古い値は失なわれる。またプログラム中の空白・改行は無視される。

3° 名前、数、変数及び comment

3.1 名前 (identifiers)

変数、名札、手続、関係指示子、スイッチに対して他と区別するためにつけられるもので、大文字に始まり場合によつてはその後大文字や数字がつづいたものである。

(ALGOL-60 では名札に対して符号なしの整数も許されている)

名前は最初の 6 文字しか区別されない。(ALGOL-60 では別段このような制限はないし、また一般に小文字の使用も認められている)

(例) A, B15, BAT4D

3.2 数 (numbers)

数は数字 + $10^{\cdot}$ が組み合わされて数値を表現するもので $10^{\cdot}$ との少くもどちらか一方を含む実数 (real) とこれらの何れをも含まない整数 (integer) とに分れる。

数については次の 3 つの制約がある。

1. 小数点で終つてはならない。
2. $10^{\cdot}$ のべきは整数に限る。
3. $10^{\cdot}$ の直前に小数点があつてはならない。

また整数は 12 ケタ、実数は $\pm 10^{77}$ 以内でなければならない。

(ALGOL-60 には別段この制限はない)。

これらの数で表現された算術定数は TRUE, FALSE の論理定数と同様式の中に直接記入して用いることもできるほか、読込命令に対応させてデータテープ (又はカード) より読み込ませることもできる。論理定数の読込みは T 又は F の字 (後に何を書いてもよい) をカンマで切つて読み込ませる。

3.3 変数 (variables)

変数には単純変数と添字付変数がある。おののおに算術変数と論理変数があり何れも宣言して、しかも代入又は読込によつて値を与えてからでなければ使用できない。

3.3.1 単純変数 simple variable

名前をそのまま用いる。 (例) A, B15

3.3.2 添字付変数 (subscripted variable)

名前の後に角カッコで包んだ添字 (2 次元以上のときはカンマでつないだもの) をつけて用いる。

(例) A [4] , B15 [1, 4]

添字には後でのべる式を用いてよい

(例) A [I+4] , B15 [1, if P=Q then 4
else 5]

3.4 comment

計算機と無関係にプログラムの情報伝達のために次の処置をすれば自由な記述が許される。

1. begin 及び ; に続いて comment と書いた場合は ; がくるまでの間何を書いてもよい。
2. end の後は次の ; がくるまでの間 end else ; 以外の何を書いてもよい。

(例) begin comment TEST;
; comment END OF SUMMATION ;
end of N cycle ;

4° 宣言 (declaration)

宣言には型宣言、配列宣言、手続宣言、関数指示宣言、スイッチ宣言、名札宣言があり、名札宣言以外はすべてそのブロックの先頭の begin に続く宣言部に於て (命令に先立つて) 行なわれ、すべての宣言はそのブロック (その宣言部の直前の begin とこれに対応する end によつてくくられた部分) に対し局所的である (すなわち、そのブロックの外では意味を持たないし、もし外と同じ名前が使われたら中の方の意味がブロック内では優先する)。

4.1 型宣言 (type declaration)

すべての単純変数は型宣言を行つてからでなければ使用してはならない。

同じ型の宣言はカンマによつて接続出来、その宣言が終れば必ず

セミコロンで区切る。

型には実数、整数、ブールの3つの型がある。

実数は少くとも小数点が₁₀のべきを含んだ数に対応し、整数はその何れをも含まぬもの、ブールはTRUE, FALSE という論理定数に対応するもので夫々 real, integer, Boolean という型宣言子を先頭に名前を書く。

名前として許されるものは何を用いてもよいが、同じ名前を同じブロック内の変数、名札、手続、関係指示子、スイッチにつけてはならない。

4.2 配列宣言 (array declaration)

すべての添字付変数は配列宣言を行なつてからでなければ使用してはならない。コンマ、セミコロンの使い方は型宣言の場合と同じであるが、同じ添字の境界を持つ配列の名前もコンマでつないでよい。

宣言子は real array, integer array, Boolean array であるが単に array と宣言されたときは real array と解される。

コロンでつないだ1組の整数値は添字の下限：上限を示しており
下限 ≤ 上限 でなければならない。

(ALGOL-60 は整数値でなくても算術式(後述)を認めている)。

(例) array A [1:5, -3:10], BA, C [1:40];

4.3 名札宣言 (label declaration)

ある命令に対して名札を立てたいときは、その命令の直前でレーベルの名前を書きその後には：をおけばよい。名札はいくら重ねて置いてもよい。

(例) LI: STOP: go to NEXT;

4.4 スイッチ宣言 (switch declaration)

いくつかの名札に最初から番号をつけてその何番目かを指定したいときはスイッチ宣言を行なえばよい。

スイッチ宣言は switch と書いた後にスイッチの名前を書き := をおいた後、指示式(後述名札もその1種)をつけたい番号順(最初から1, 2 …… と解される)にカンマでつなぎセミコロンで終ればよい。

この場合スイッチの構造はその要素に 1, 2, 3 と番号のついた一種の配列であると考えられることになり、S というスイッチの J 番目は S [J] となる。

```
(例)  switch SA:=L1, STOP, if P=1 then END
        else L3;
        switch P:=SA [ I ], L5;
```

4.5 手続宣言 (procedure declaration)

同じような手続で計算を行なう部分は何箇所あるとき、そのたびに繰り返して殆んど同じプログラムを書くことを避け、この似た手続のうち変る部分をパラメタによつて指定してこの手続全体につけた名前を呼び出して実行させるためには、まづこの名前と内容を宣言しておくかなければならない。

その形は

```
procedure 名前 ( 形式的パラメタ部 );
    <value part>;
    <specification part>;
    <procedure body>;
```

である。

形式的パラメタ部はパラメタを使用しないときは不用であり value part は value 指定するパラメタがないときは不用である。

specification part は形式的パラメタのすべてについて指定を行う必要がある。(ALGOL-60 では value 指定してないパラメタについては不用)

手続の実行に当つて値を外から受取つてくる変数は value part で value 指定を行なう。何回も手続本体の中で使うパラメタで、値を外から持つてくる(これを値と呼ぶ call by value という)ものは value 指定したほうが演算時間が短くなる。値で受取らず名前と呼ばれた(これを call by name という)パラメタは、いちいちその番地を頻りに情報を受取りに行くために演算時間がかかるので、

値で呼ぶことのできるパラメタは value 指定したほうがよい。

array は値で呼ぶことができない。

値で呼んではならぬパラメタとしてはこのほかにその手続の結果値が得られるものなどがある。

明細指定部 (specification part) で用いる明細指定子 (specifier) としては real, integer, Boolean, real array, array, integer array, Boolean array, label, switch, procedure, real procedure, integer procedure, Boolean procedure, string がある。

明細指定は宣言とは別物で、形式的パラメタはこの手続に関し局所的である。たとえば明細指定の array は宣言とちがつて名前だけを書く。

この手続宣言に対応して使用する場合には手続命令 (後述) による procedure body において global (非局所的) な量を使用するときは既に宣言されたものでなければならない。

4.6 関数指示子宣言

ある手続の結果として1つの数値又は論理値が得られるときは、その結果を手続につけた名前のところにに入れておき、この関数指示子 (function designator) を式の中で直接使用することができる。

これに対応して関数指示子宣言が必要であるが、SIN, COS, ARCTAN, LN, EXP, SQRT, SIGN, ABS, FLOAT, FIX は無宣言で使用してよい。

SIN, COS, ARCTAN はその後のパラメタに任意の算術式を用い答はラジアン単位で得られる、LN, EXP も任意の算術式を用い答は real で得られる。

SQRT は非負であるような算術式を用い答は同じく real, SIGN はシグナムと読み任意の算術式に対しそれが正なら+1、0なら0、負なら-1が結果として得られるようになっている。

ABS は任意の算術式に対しその型の絶対値が得られる。FLOAT は integer を real に、FIX は real の算術式に対し結果を切りす

て integer にしたものとするそれぞれ関数指示子である。

(ALGOL-60 はすべて小文字で PLX に対応して entier アンテ
イエがありそれを越さぬ最大の整数をとる)

これと似た働きをする関数指示子は関数指示子宣言を行うことによ
つて任意に作り出すことができる。やり方は手続の場合と同じである
が、宣言子が

```
real procedure, integer procedure, Boolean procedure
procedure
```

という形をとる点と procedure body の中で少くも 1 回自分自身の名
名前に対し

名前 := 算術式;

という代入命令がある点である。

また使用する際は手続命令と違い算術式やブール式の中に直接書き
こまれる。

4.7 external 宣言及 code

同じ手続宣言や関数指示子宣言が違ったプログラムに何度も現れる
とき、いちいち宣言をくり返して書かないで計算機にその宣言自身を
記憶させておいて、使用するプログラムの宣言部で external 宣言
で名前だけを書いて使用することができる。

そのため機械語に訳された内容を磁気テープに格納するには

```
attach procedure 名前 (形式的パラメタ部);
|
|      <value part>;
|      <specification part>;
|      <procedure body>;
|
end ;
```

というテープを作つてかけてやればよい。関数指示子宣言なら

```
attach real procedure ..... 以下同じ
```

```
|
```

```
end ;
```

などとすればよい。

使用に当つては先づ宣言部で

```
external 名前、名前、……；
```

```
external integer 名前、名前、……；
```

等とするだけで使用してよい（関数指示子は type をつける点で異つてゐる）。

実際には external procedure は attach された順に格納される。

（ALGOL-60 にはこのような external 扱いのやり方はない）

また機械語（8進）で書かれたプログラムを接続するには

procedure body の中で

```
code < 8 進数 14 ケタ 単位の機械語 >；
```

とすれば external procedure の body では接続可能である

（ALGOL-60 にはこの規定はない）。

4.8 手続宣言、関数指示子宣言における形式的パラメタ

手続宣言、関数指示子宣言はそれ自身ブロック扱いとなる。従つて形式的パラメタはその宣言に関し局所的である。また明細指定が型宣言等によく似ているため局所的でない変数と同じ名前を使つたとき、既に global に宣言してしまつた様に錯覚しやすい。従つて形式的パラメタと同じ名前を実パラメタに使うときはその実パラメタに用いた名前が global に宣言してあるかどうかは注意を要する。

変数は value 指定してあれば real integer の型が異つても良いが配列の名前については明細指定部では宣言と違つて添字の限界に触れず名前だけを明細指定することと array というと real array と解されるので実パラメタに使う予定のものが real か integer かについては注意しなければならない。

以下特に形式的パラメタとしての procedure, < type >

procedure, string, label, switch について注意すべき点に触れておく。

4.8.1 procedure 及 < type > procedure

同じレベル又は外側のブロックで宣言された procedure を形

式的パラメタとして指定するときは適当な名前をパラメタ部に記入しこの名前を明細指定すればよい。勿論 procedure body の中で使うときはその宣言がパラメタ付きならば適当な実パラメタを書き込んで使用する。

4.8.2 string

たとえば印刷手続きで印字内容を勝手に変えてその右に数値を打ちたいときその印字する部分を代表するものとしてパラメタの string を用いる。命令中の実パラメタとしては対応する場所に印字したい文字を ' ' につつんで書いておけばよい。

4.8.3 label 及 switch

条件によつて手続本体から飛び出してある名札又はスイッチ要素を選択したいとき形式的パラメタとしてレーベル又はスイッチ名を指定する。procedure body の中では go to 命令としてその名前が使われ、命令中の実パラメタとしてはその時に選ばれるレーベルやスイッチ名が書かれる。

4.8.4 value 指定

実パラメタ部に式を書く予定の形式的パラメタは一般に value 指定しておかなければならない。但し procedure body の中で代入命令の左辺に來ても誤りとならない array 要素の場合は別であるが、実パラメタ部に 4.5 とか $P * Q$ といった式を書く予定があると、これらが実行に當つて代入命令の左辺に現れることになつて誤りとなる。

一般には array の名前、その procedure の実行の結果値の得られるものを除いて value 指定しておく方が flexible なプログラムを作ることができる。従つて procedure body の中で代入命令の左辺に現れる予定の変数は value 指定してはならない。(NEW H の場合 array 要素は式であつても call by name の形式的パラメタに対応した実パラメタとして書くことができる)

5° 式 (expression)

5.1 算術演算子 (arithmetic operator)

$+$ $-$ $*$ $/$ $\div$ $\uparrow$ はその両側に変数、数、関数指示子または算術式自身あるいはこれらを適宜丸カッコでくくつたものなどをおいて算術式を構成する。ただし $+$ $-$ に限つてその前に何もない場合が認められる。

算術演算子の優先順位は次の通りである (丸カッコを含む)

1.	丸カッコ	()
2.	べき	$\uparrow$
3.	乗除	$*$ $/$ $\div$
4.	加減	$+$ $-$

同順位のものの中では式の左側に書かれたものの方が優先する。

$\uparrow$ については $A < 0$ ならば $A \uparrow B$ は B が integer のときのみ定義される。また $A = 0$ のとき $B \leq 0$ ならば定義されない。 $2 \uparrow N \uparrow K$ は $(2^N)^K$ であり、 $2 \uparrow (N \uparrow K)$ が $2^{(N^K)}$ となる。

$\div$ は整数同志の割算に用い、その結果は絶対値の切捨を行つたものに符号を付したものとなる。たとえば

$$-5 \div 2 \rightarrow -2$$

でこれを利用して A が N で割り切れるかどうかをチェックするのに

$$A \div N * N = A$$

かどうかを調べるやり方がある。

一般に、 $*$ $+$ $-$ はその両側の式がともに整数のときのみ整数となり ($/$ はその両側がともに整数のときも結果は実数) 他はすべて結果が実数となる。

5.2 関係演算子 (relational operator)

$\geq$ $=$ $\leq$ $<$ $>$ という関係演算子はその両側に単純算術式 (後述) がくることによつて関係 (relation) を構成しブール式 (後述) の構成要素の一つとなる。関係演算子は算術式を狭んで2つ以上連続して用いてはならない。

関係演算子は算術演算子に優先されるが論理演算子に優先する。

5.3 論理演算子 (logical operator)

論理演算子には $\neg$ (not) $\wedge$ (and) $\vee$ (or) (imply)

$\equiv$ (equivalent) があり、この順序に優先する。

従つて各演算子等を通しての優先順位は次表の通りである。

1.	()
2.	$\uparrow$
3.	$*$ $/$ $\div$
4.	$+$ $-$
5.	$>$ $\geq$ $=$ $\leq$ $<$ $\neq$
6.	$\neg$
7.	$\wedge$
8.	$\vee$
9.	$\supset$
10.	$\equiv$

$\neg$ 以外の論理演算子はその両側にブール定数、ブール変数、関係、またはブール式を並べることによつて ($\neg$ のみはその後にくるブール定数、ブール変数、関係またはブール式を否定する働きを持つ)、さらにブール式を構成する。

A、BをBooleanと宣言されたブール変数とすれば

$\neg A$: AがTRUEのときFALSE、逆にFALSEのときTRUE
 $A \wedge B$: A、BともにTRUEのときのみTRUE
 $A \vee B$: A、BともにFALSEのときのみFALSE
 $A \supset B$: AがTRUE且つBがFALSEのときのみFALSE
 if A then B 又は $\neg A \vee B$
 $A \equiv B$: A、BともにTRUEまたはFALSEのときのみTRUE
 $A \wedge B \vee \neg A \wedge \neg B$

$\wedge$ はAndと関連して記憶するとよい。

$\neg$ 以外の演算子を2つ以上続けて使用することはできない。

(例) 誤 $A \wedge \vee B$

5.4 算術式 (arithmetic expression) $\overbrace{AE}$

算術式は算術定数、算術変数、算術関数指示子、カッコ算術演算子、
if then else によつて構成され数値を計算する規則を与えるもの
であり、次の2種がある。

単純算術式 (simple arithmetic expression) $\overbrace{SA}$

if then else を含まぬかあつても丸カッコに包まれた
 $\overbrace{AE}$

条件付算術式 (conditional $\overbrace{AE}$)

if $\overbrace{BE}$ then $\overbrace{SA}$ else $\overbrace{AE}$

条件付算術式は if then else の3つを必ず含んでおり、
then と else の間は $\overbrace{SA}$ に限る。もしここに条件付算術式を入れた
ければ、それを丸カッコで包んで $\overbrace{SA}$ に直せばよい。

また条件付算術式を式の中で用いるときは丸カッコで包まないと意
図が不明になることがあるので注意を要する。丸カッコは対応してさ
えいれば余分にあつてもさしつかえない。

算術式は次のようなものが考えられる。

1. 算術定数 0.123
2. 算術変数 SUM
3. 算術関数指示子 $\overline{COS}(X+3*Y)$
4. $1\sim3$ を算術演算子で結んだり $0.123 * SUM + A(I) - 1$
カッコを付加したりしたもの
5. if $\overbrace{BE}$ then $\overbrace{SA}$ else $\overbrace{AE}$ if $A \geq B$ then (if $P \geq 0$
then $P * (Q + 1)$ else
 $P * (Q - 1)$) else $P * Q$
6. $1\sim5$ をカッコに包んだもの
(これを $1\sim4$ と共に $\overbrace{SA}$ という)
7. $1\sim6$ を算術演算子で結んだり
if then else やカッコを
付加したもの

5.5 ブール式 (Boolean expression) $\overline{BE}$

ブール式はブール定数、ブール変数、ブール関数指示子、関係やブール式自身を $\neg$ 以外の論理演算子で結んだもの、さらにこれらに $\neg$ を付加したりカッコでくくつたりして論理値を計算する規則を与えるもので、次の2種がある。

単純ブール式 (simple $\overline{BE}$) $\overline{SB}$

if then else を含めぬかあつても丸カッコで包まれた $\overline{BE}$

条件付ブール式 (conditional $\overline{BE}$)

if $\overline{BE}$ then $\overline{SB}$ else $\overline{BE}$

条件付ブール式は if then else の3つを必ず含んでおり、then と else の間は $\overline{SB}$ に限る。もしここに条件付ブール式を入れなければ丸カッコに包んで $\overline{SB}$ に直せばよい。丸カッコの使い方は $\overline{AE}$ の場合と全く同様である。

ブール式は次のようなものが考えられる。

1. ブール定数 TRUE 及び FALSE
2. ブール変数 CR BA
3. ブール関数指示子 RUN (P, Q)
4. 2個の $\overline{SA}$ を関係演算子で結んだもの $A > 3.5$
5. 1～4を論理演算子で結んだり $\neg$ や丸カッコを付加したもの $TRUE \vee BA > 3.5 \wedge CR$
6. if $\overline{BE_1}$ then $\overline{SB}$ else $\overline{BE_2}$ if CR then (if $A > 3.5$ then $D=C$ else $\neg$ RUN (P, Q)) else $TRUE \vee BA$
7. 1～6をカッコで包んだもの (1～5と共に $\overline{SB}$ という)
8. 1～7を論理演算子で結んだり if then else やカッコを付加したもの

5.6 指示式 (designational expression) DE

指示式とは名札を得る規則であり、名札、スイッチ指示子、丸カッコ及び if then else がその構成要素で次の2種がある。

単純指示式 (simple designational expression) SD

if then else を含まぬかあつても丸カッコで包まれた
DE

条件付指示式 (conditional DE)

if BE then SD else DE

後者についてはAE, BEの場合と全く同様である。

指示式は次のようなものが考えられる。

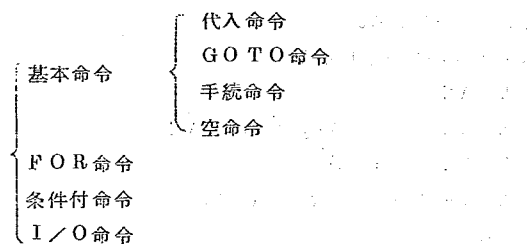
1. 名 札 L1
2. スイッチ指示子 S { I }
3. if BE then SD else DE if A=B then L1
else S { I }
4. 1～3 をカッコに包んだもの
(1～2と共にSDという)
5. 1～4 に if then else や
カッコを付加したもの

6° 命 令 (statement) S

命令は計算機に一つの操作を実行させる単位であつて、通常 (GOTO 命令などによつて順序を変更しない限り) その書かれた順序に実行される。

1つの命令と他の命令の区切りは ; によつて行ない誤りを少なくするためには複合命令等も含めて命令の末尾は else の直前を除いて必ず ; を入れるとしておいた方がよい (endの直前は不用であるが空命令を立てたと考えれば ; があつて差し支えないので一部をとつて他のプログラムにはめ込む時のことを考えると、すべての命令の末尾には ; を付ける原則にしておいた方がよい)。

命令は次の種類に分れる。



基本命令と I/O 命令を無条件命令 (unconditional S US) という。

6.1 代入命令 (assignment S)

$$\begin{aligned} \underline{V_1} &:= \underline{V_2} := \dots := \underline{AE}; \\ \underline{V_1} &:= \underline{V_2} := \dots := \underline{BE}; \end{aligned}$$

という形をとり、:= の右端 (右辺という) の式の結果を左辺 (同じ型ならばいくつあつてもよい) の変数に代入させる命令で左辺は変数 (単純変数と添字付変数) に限る。

右辺と左辺は integer real のどのタイプであつても左辺の型に合わせて代入され、左辺が整数のときは四捨五入を行なう。

新しい値が代入されたら古い値は失なわれる。

なおプログラミングするときに分っている数値は勿論そのまま AE に書き込んでよいが、計算実行時に外から知らせてやる必要のある数値は I/O 命令によつて読み込めばよい。

:= は代入命令だけでなく次の場合にも用いられる。

スイッチ宣言 switch I := DE₁, DE₂, ... ;

for 命令 for I := <for list> do S ; (後述)

関数指示子の body の中で

<関数指示子の名前> := AE ; (又は BE)

:= は再帰的であつて左右両辺に同じ名前が現れても勿論差しつか

えない。

6.2 FOR 命令 (for S) FS

命令 S をくり返し実行させたい場合に

for <variable> := <for list> do $\bar{S}$;

という形を用いる。

<for list>としては

1. $\bar{AE}$
2. $\bar{AE}_1$ step $\bar{AE}_2$ until $\bar{AE}_3$
3. $\bar{AE}$ while $\bar{BE}$
4. 1～3 をカンマでつないだもの

がある。

do の後は1つの $\bar{S}$ しか実行しない。従つて2つ以上の $\bar{S}$ を実行させたいときはこれらをbegin end でくくつて1つの複合命令 (compound $\bar{S}$) にしてやらなければならない。do の後はいつもbegin end でくくる習慣をつけるとエラーが少なくなる。begin end は丸カッコと同じく対応してさえいれば、余分にあつても何等害はない。

(例)

```
for A:=1 step 2 until 10, 15 step 5
  until 50, 60 step 10 until 100 do  $\bar{S}$  ;
for B:=2.5, 3*B while B ≤ 1500 do  $\bar{S}$  ;
```

6.3 GOTO 命令 (go to $\bar{S}$)

あるステートメント (それは複合命令やブロックでもよい) に名札がつけてあれば

```
go to  $\bar{DE}$  ;
```

という命令によつてその名札のところにコントロールを変更できる。

$\bar{DE}$ であるから名札の名前であつても、if then else による条件付指示式であつてもスイッチ指示子であつてもよい。

```
go to  $\bar{S} [J]$  ;
```

に対しJがスイッチ宣言にない範囲のものであるときは誤りとなり (ALGOL-60 では空命令扱いとなるがこれはかえつて何処が悪かつたか分りにくい)、Jのところに算術式が来て結果がreal のとき

は自動的にFIXされる。

6.4 条件付命令 (conditional S)

条件付命令には次の3種がある。

- | | | | | |
|----|----|----|------|-------------|
| 1. | if | BE | then | US; |
| 2. | if | BE | then | US else S ; |
| 3. | if | BE | then | FS; |

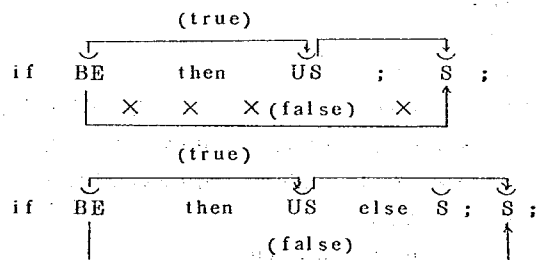
US とは無条件命令であつて命令中条件付命令及びFOR命令以外の
ものであり代入命令、GOTO命令、空命令、手続命令、I/O命
令、複合命令、ブロックなどがある。FS を US と切り離した理由
は

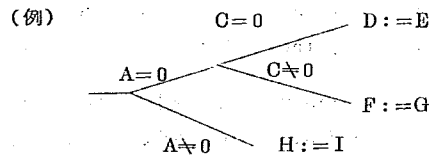
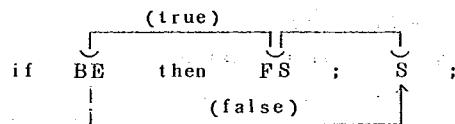
if BE then for I:=<for list> do if BE
then US else S ;

というFSのdoの後にifが来たときelse S がどちらのif に対
応するものであるか分からないためかとも考えられる。しかし NEW
ALGOL-Hのコンパイラはelseは一番近くのifに対応するものと
解釈して処理することになっている。

このような不明確さを避けるにはdoの後のSは、たとえ1つであ
つてもbegin とendでくくることにしておくか、或いはthenの後
のSをbegin とendでくくることにすればよい。

else の直前には絶対に;をつけてはならぬという点も注意を要す
る。条件付命令による処理の進行は下図のようなものになる。





```

if A=0 then begin if C=0 then D:=E
                else F:=G end else H:=I;

```

6.5 空命令 (dummy S)

何も実行しないという命令がありこれを空命令という。勿論次にはそれに続く命令を実行する。

ある条件が成り立てば終りにしたいときに空命令に対して適当な名札をつけて、その次に終りの end を持つてくるという使い方も空命令の利用方法である。

```

if A=0 then go to END ;
S1 ;
S2 ;
END :
end ;

```

6.6 手続命令 (procedure S)

同じような手順をくり返して用いるときは変る部分を形式的パラメタとした手続宣言をしておけば、本文中で宣言中の形式的パラメタに対応して実パラメタを伴った手続の名前を書いて手続命令とすることができる。その形式は

$I(\overline{AP}, \overline{AP}, \dots, \overline{AP}) ;$

であつて、勿論パラメタのない手続はその名前だけ書けばよい。ただしここで混同してならないことはたとえ形式的パラメタ (FP) と同じ名前を AP に用いたとしても FP は手続宣言中の局所的なものに過ぎないから、AP を用いるブロック内では必ず宣言してなければならない点である。

AP は FP とその個数、型、種類等が一致していなければならない。(ALGOL-60 では必ずしもその必要はない)

手続命令が出ると手続宣言における FP に対し、いま書きこまれた AP がそれぞれ value 指定のあつたものは値を、その他は名前をやりとりして手続を実行する。なお、手続中で代入命令の左辺に来る FP に対応する AP に変数以外のものを書いてはならない。

関数指示子は式の中で用いる。AP に関する注意は手続の場合とほとんど同様であるが、この場合は 1 個関数であるからパラメタは全部 value 指定できる筈である。

7° I/O 命令 (input - output S)

ALGOL-60 には I/O に関する何等の規定もない。

NEW ALGOL-H では次の I/O 命令を用いる。

7.1 読み込み

テープ (又はカード) より順次数値を読み込み、対応する名前のところに格納するのは

```
READ ( I1, I2, ..., In );
```

とする。integer real 何れであつても名前の型に合せて変換される。

データは integer 及 real は

4, 5, 3.32, ... ; ① ①

Boolean は

F, T, F, F, ... ; ① ①

とカンマでつなげばよい。

電算運営資料

若し 1 2.5 というデータが 1 0 個連続するときは

1 0 * 1 2.5,

とすればよい。またカンマだけがあるときは 0 が読み込まれる。

読み込みの 1 時停止は

halt ; ⑩⑩

で⑩はテープマークの意である。

文字の読み込みは = Y O K O H A M , = A , = T O K Y O , = , ...
等とする。= の後は 7 文字まで文字として読み込むので 7 文字 1 組に
しなければならない。= の後に何もなければブランクを読み込む。た
だし文字の間で空白を指示したいときは = | # T O K Y O , などと #
を用いる。

7.2 配列読み込み

3 次元の配列 A で N 行 M 列のマトリクスに対しては行なりに入つて
いるデータに対応して

```
READARRAY(A, 1, N, 1, M);
```

とする。何次元でもよく -5, 4 などと任意の bound を指定できる。
尚宣言された通りの大きさの配列を読み込むなら

```
READ(I);
```

でもよい。

7.3 文字の印刷

××...× という文字を印刷準備するには

```
PRINT("××...×");
```

とする。内側の ' ' につづまれたものを、そのまま印刷準備する。外
側の ' ' はフォーマットを示すものである。

n 回くり返したいときは

```
PRINT('n('××...×')');
```

でよい。

7.4 スペースの印刷

スペース（ブランク）を n 個入れたければ

```
PRINT(' n B ');
```

でよい。

7.5 復帰改行

今迄の情報を 1 行に印字し n 回改行したければ

```
PRINT(' n / ');
```

とする。120 字なり 135 字なりの LP（ラインプリンタ）が、ある情報が次に入ることによつて 1 行分をオーバーするとその最後の情報の前までを 1 行分として印刷し、最後の情報は新しい行の先頭に入つて待機することになる。

また今迄の n 回という n は符号のない整数なら何を用いてもよく、後述の X を用いて変数による指定も可能である。

7.6 用紙の先頭よりの印刷

LP 用紙の先頭位置から印刷したいときは

```
PRINT(' ↑ ');
```

であり n 枚おくつてから先頭より 5 行目より印刷したければ

```
PRINT(' n ↑ 5 / ');
```

となろう。

7.7 簡単なフォーマット指定

real integer Boolean タイプの $\overbrace{AE}$ 又は $\overbrace{BE}$ を一様な形式で印字したいときは

```
PRINT(' R' ,  $\overbrace{AE}$  );
PRINT(' I' ,  $\overbrace{AE}$  );
PRINT(' L' ,  $\overbrace{BE}$  );
```

とすればよい。

$\begin{array}{c} \text{'R'} \quad \square\square\pm, \underbrace{\times\times\cdots\times\times\times}_{9\text{ケタ}} \cdot \underbrace{10\pm\times\times}_{2\text{ケタ}} \end{array}$
 1.7 文字分
 (ゼロサプレス)

$\begin{array}{c} \text{'I'} \quad \square\square\pm \underbrace{\times\times\cdots\times}_{13\text{ケタ}} \end{array}$
 1.6 文字分
 (ゼロサプレス)

$\begin{array}{c} \text{'L'} \quad \square\square\text{TRUE} \\ \square\square\text{FALSE} \end{array}$
 } 7 文字分

R と I との間ではフォーマット指定の型に合せて変換される。

これらは整数個のくり返しを指定してもよいし、/B ↑ とくみ合わせてもよい。

ただし R、I、L は次の F、D、Z、A、S と同じ様にカンマで区切る必要がある。

(例) integer B1 の右にスペースを2個入れ、real A1,
 C2, RN を用紙の先頭から印刷し、A1 の左に DATA= と
 いう字を入れ1行に印刷したいとき
 PRINT('↑I2B, 'DATA='3R/', B1, A1, C2,
 RN);

7.8 詳細なフォーマット指定

7.8.1 算術式の印刷

Z ゼロサプレスする桁

D ゼロサプレスしない桁

—又は+ 符号を付ける要求

. 小数点

10 10 のべき

を用いてたとえば A は real で小数点上4ケタ、下3ケタ、B は integer で8ケタ、C は $\pm, \times\times\cdots\times_{10}\pm\times\times$ の形で7ケタ印刷し、何れも間に2個のブランクを入れ1行に印刷するときは

```
PRINT(' -4Z, 3D, 2B-8Z, 2B-, 7D10-ZD/' ,
      A, B, C);
```

となる。これらは3(-4Z, 3D)とくり返させることもできる。

7.8.2 論理式の印刷

TRUE または FALSE と印字させたいとき

```
PRINT(' 5F' , BE);
```

で、もし 'F' なら T または F を打つ。1 または 0 を印刷させたいければ

```
PRINT(' P' , BE);
```

でよい。

7.9 アルファモードの印刷

I のアルファモードの情報をその文字内容の通り7字印刷準備するには

```
PRINT(' 7A' , I);
```

とする。

アルファモードは読み込んでもよい(=を用いる)が、プログラム中で代入するには、たとえば integer K に P39A という文字を代入したければ

```
K := EQUIV ( P39A );
```

としておいて

```
PRINT(' 4A' , K);
```

でよい。1語には7文字以上入らないことは注意を要する。

7.10 マグネティックテープ

array A を MT の n 番地から格納したければ

```
PUT ( n , A );
```

これをBに入れて使用したければ

```
GET ( n , B ) ;
```

でよい。

このほかに

```
MTWRITE (<unit> , <format string> , <list> ) ;
```

```
MTREAD (<unit> , <format string> , <list> ) ;
```

も使用できる。

7.11 変数による個数制御

フォーマット中の符号なし整数はすべてXによる制御が可能である。

たとえば↑をN回、RをM個、IをP個、1をQ個でその順にフォーマット中に書くなら

```
REPX ( N , M , P , Q ) ; としておいて
```

```
PRINT ( ' X↑XR , XIX / ' , I , ... , I ) ;
```

でよい。

8° 特殊な取扱い

8.1 アルファモード

ALGOL-60 にはアルファモードに関する規定はないが、NEW ALGOL-H ではデータテープ上=を先頭にした7文字以内の情報を integer タイプの変数に読み込ませたりプログラム中で integer タイプの変数に対し

```
I := EQUIV ( × × ... × ) ;
```

という形でアルファモードの情報(7文字以内)を代入し、

```
PRINT ( ' 7A ' , I ) ;
```

として印刷準備することができる。

代入読み込みは7文字以内であるから8文字目からは次の変数に入れなければならない。積極的にブランクを入れるには#を用いればよい。

8.2 条件付算術式及び条件付命令

```
条件付算術式  if BE then SA else AE
```

のほかに on <unsigned integer> then SA else
 AE
 条件付命令 if BE then US;
 if BE then US else S;
 if BE then FS;

のほかに
 on <unsigned integer> then US;
 on <unsigned integer> then US else S;
 on <unsigned integer> then FS;

がNEW ALGOL-Hに作られた(これはALGOL-60にはない)
 コンソール上の1~10のボタンに対応して、これが押してあつたら then 以下に行かせるためのもので、チェック計算の時に計算を簡略化したりする時に利用できる。

9° プログラムエラーのチェック

プログラムに文法的エラーがあれば機械がエラーメッセージを打つてくるが一応次の点についてチェックしてから機械にかけることが望ましい。

9.1 算術演算

1. 掛算の*をおとして(A+B)Cなどとしていないか?
2. 割算の分母に0が入ってくるようになっていないか?
 (代入や読込によって値を与えてない変数をいきなり使っていないか)
3. A↑-5とか3/-4.8などと2つ以上の算術演算子を並べて使っていないか?(カッコを用いる)

9.2 変数

1. 未宣言の変数を使っていないか?
 (特に手続命令などの実パラメタが形式的パラメタと同じ名前するときなど)
2. 同じブロック内で変数、名札、手続、関数指示子、スイッチ等の名前に同じものを使っていないか?

電算運営資料

3. 代入又は読込によつて値を与えないで変数を使つていないか？

2.3 デリミタ

1. デリミタの綴りに誤りはないか？
2. begin end, (), [], ' ' などが対応しているか？

2.4 名 前

1. O と $\bar{O}$ と D を間違えてないか？
2. l と 1 と I と / を間違えてないか？
3. r と v と V を間違えてないか？
4. (と O を間違えてないか？
5. 2 と Z を間違えてないか？
6. 5 と S を間違えてないか？

2.5 式

1. 条件付算術式、ブール式、指示式は if then else の3つともを含んでいるか？
2. 条件付の式で then の次に if がくるときは、if 以下の式を () でつつんであるか？

2.6 命 令

1. 命令は else の直前を除きすべて ; で終つているか？
特に begin end でつつんだ複合命令の場合に end の後の ; を落していないか？ (end の直前では不用であるが end の直後と間違いやすいので命令はすべて ; で終ることにしておいた方が誤りが少ない。空命令があると考えれば end の直前に ; があつてもさしつかえない)
2. 代入命令の左辺の変数の型は皆同じものか？
3. 条件付命令で then の後に if がくるときは if 以下の命令を begin end でつつんであるか？
4. for 命令の do の後のくり返すべき命令は最初のセミコロン (begin がくるときは対応する end の後のセミコロン) までであるがそれでよいか？
5. for 命令の中で初めにもどつて初期値からやり直していることは

ないか？（初期値代入が do の後に入っている場合）

6. for 命令のコントロール変数について、指定通りくり返し計算が終了してから次の命令の中でこのコントロール変数の値を利用していないか？
7. for 命令の do の後のくり返すべき命令の中に入っている READ 命令に対応してデータテープが用意されているか？
8. procedure 中で for 命令に用いる I、J 等のコントロール変数を procedure 中で宣言しなかつたために本体の I、J と混同を起すことはないか？
9. end の後の ; を忘れたために次の命令が comment 扱いとなつて読みとばされるおそれはないか？
10. comment は ; で終っているか？
11. for 命令の中で毎回の答を格納し忘れてないか？

{4} FORTRAN(IV) と ALGOL(NEW-H) の比較

中 村 慶 一

はじめに

FORTRAN と ALGOL とは非常によく似ている。というよりは1956年 IBM 704 用に開発されたいわゆる FORTRAN II の特長もとり入れ、国際共通語として1958年、1960年、1962年の改訂を経た改訂 ALGOL-60 の誕生の経緯ならびに逆に ALGOL の特長を取り入れた FORTRAN IV の成立までのいきさつを見れば似ていない方が不思議な様なものである。しかし詳細に見ると IBM の機械の user の便宜を主体とした見方と、最初から一般化を考えて文法体系を考えた言語との差は随所に見られ習得ならびに日常の使用に当つての容易さや他人の書いたプログラムを見て理解すること（人間対機械でなく人間対人間の communication）の容易さにおける差異はやはり存在する様に感ぜられる。

土木研究所に於て既に450人のプログラムの養成を終り、毎週何日かの終夜運転を必要とする状態まで起すようになったが、これを FORTRAN で行っていたら果してこの様に急速な普及を見ることができたかどうかは、はなはだ疑問であると考えられる。

しかしその対比について質問を受ける機会も多いので同種のを並べて比較を試みてみたい。

以下 FORTRAN を 'F'、ALGOL を 'A' と略記する。

1° プログラミング上の原則

- 1.1 プログラムは statement の集まりから成り立っており、特にその実行順序が変更されない限りはその記述順に実行される。しかし 'F' では1つの statement（以下 S と書く）は1行に 'A' では ; で区切つて示される。
- 1.2 statement は 'A' に於ては begin end でくくつて1つの statement とみなすことができるが 'F' にはこのような statement をひとまとめにする機能はない。
- 1.3 変数の型、配列の大きさなどを指定するために 'F' には示様

statement 'A' には declaration がある。

1.3.1	REAL <i>IA, LB, MN</i>	real <i>IA, LB, MN</i> ;
1.3.2	INTEGER <i>AB, XY</i>	integer <i>AB, XY</i> ;
1.3.3	LOGICAL <i>A, GOOD</i>	Boolean <i>A, GOOD</i> ;
1.3.4	DOUBLE PRECISION <i>P, Q</i>	—
1.3.5	COMPLEX <i>X, Y</i>	—
1.3.6	DIMENSION <i>A(15), B(50)</i>	array <i>A(1:15),</i> <i>B(1:50)</i> ; integer array <i>C, D(1:20, 1:30)</i> ;
1.3.7	—	Boolean array <i>S(1:10, 1:5,</i> <i>1:100, 1:14)</i> ;
1.3.8	COMMON <i>A, B, S</i>	—
1.3.9	EQUIVALENCE(<i>A, X</i>)	—

注1 'A', 'F' 共示様 $\bar{S}$ 或いは declaration($\bar{D}$) は実行可能な $\bar{S}$ の前に来なければならない。'A' では途中に begin を置けば、その後に $\bar{D}$ をおくことができ、この $\bar{D}$ はその begin に対応する end までの間でのみ有効となる。(これをブロックと云い、このため数人のプログラマの共同作業が容易となる)

注2 名前は 'A', 'F' 共文字を先頭にしてその後に必要なに応じて文字又は数字が続いたものである。

注3 'F' は特に指定のないときは *I, J, K, L, M, N* を先頭とするものは integer 他は real と解され *I ~ N* を先頭とする real タイプの変数を必要とするときは示数を必要とする。'A' ではすべての名前は宣言を必要とする。

注4 論理変数は 'F' では LOGICAL, 'A' では Boolean として宣言するが論理定数は 'F' では .TRUE. .FALSE. 'A' では TRUE FALSE (ALGOL-60 では true false) である。

注5 DOUBLE PRECISION (倍精度演算) 及 COMPLEX (複素数演算) の示様は 'A' にはない。

注6 配列宣言は 'F' では A_1 より A_{15} までを $A(15)$ 'A' では $A(1;15)$ と書くが A_{20} より A_{35} のときも 'F' では $A(35)$ として 35 個取らざるを得ない。

'A' では $A(20:35)$ とする。また $A-15$ から A_{10} までは 'F' では取れないが 'A' では $A(-15:10)$ である。また 'A' では別段制限はないが 'F' では 3 次元までの配列しか認められない。従って 1, 3, 7 の Boolean array S のような 4 次元の配列は 'F' では不可能である。

注7 'A' のようなブロック構造がないため 'F' では common S によつてこのリストの先頭から何番目という順位の同じものは異った名前であつても共通扱いとなる。これによつて COMMON 領域に場所が占有される。

注8 'A' のようなブロック構造がないため 'F' では同じ記憶場所を割当てるため EQUIVALENCE によつて共通する。この場合 X は array であるとすれば B と X とは同じ場所を共用する。

1.4 I/O statement

'A' では別段規程はなく 'F' では IBM の規定に合せて format に至るまで詳細に規定してある。ここではこれから後の説明に必要な程度のみを 'F' は HITA 5020 E 用 HARP, 'A' は OKITAC 5090 H 用の NEW ALGOL H によつて説明しておく。(但し HARP は機種指定のナンバーを含むのでこれは外しておく)

1.4.1 直結の機器よりの読み込み

'F' では format 指定を必要とする。A, B で field 12 ケタの floating タイプの情報ならば

READ 100, A, B READ(A, B);

100 FORMAT(2F12.0)

1.4.2 直結印刷機による印刷

小数点以上、以下 7 ケタで field 15 ケタの情報 A, B, C を

1行に印字したとき

```
WRITE 101, A, B, C PRINT('3R', A, B, C);
101 FORMAT(1H, 3E15.7)
```

1.4.3 文字の印刷

N という integer タイプの情報4ケタを $N=$ の右に

H という real タイプの情報7ケタを $N=$ の右に

印刷するとき

```
WRITE 104, N, H PRINT('N=I, 'H='R', N, H);
104 FORMAT(1H, 2HN=I4, 2HH=E15.7)
```

2 式 (expression)

2.1 算術演算子

'F' では $+$ $-$ $*$ $/$ $**$

'A' では $+$ $-$ $*$ $/$ $\uparrow$

で $\div$ は整数割算であり 'F' では両辺整数のとき同じ結果となる。

'F' では $**$ を除いて両辺のタイプが同じでなければ誤りとなるが

'A' では両辺のタイプが異なるときは自動的に real となり $/$ の両辺が整数のときもその結果は real となる。

2.2 関係演算子

'F' では $.GT.$ $.GE.$ $.EQ.$ $.LE.$ $.LT.$ $.NE.$

'A' では $>$ $\geq$ $=$ $\leq$ $<$ $\neq$

2.3 論理演算子

'F' では $.NOT.$ $.AND.$ $.OR.$ 'A' では $\neg$ $\wedge$ $\vee$ $\equiv$

2.4 算術式 (arithmetic expression)

'F' では算術定数、算術変数 (同じ型に限る)、FUNCTION、算術演算子、カッコで構成される。

'A' では算術定数、算術変数 (real integer 何れも可、他の定数、関数指示子も同じ)、関数指示子、算術演算子、カッコの外に if then else による条件算術式を作ることでもある。

$D \leq 0$ なら $X1 = S + T$ さもなければ $X1 = -B/A$ としたければ

```

IF(D.LT.O.O) GO TO 1  X1:=if D≤0 then
X1=-B/A                S+T else -B/A;
1 X1=S+T
P=Qなら A1, i + B   さもなければ A2, i + B を S に代入したけ
れば
TF(P.EQ.Q) GO TO 2    S:=A[ if P=Q then 1
X=2                    else 2, I ] + B;
2 X=1
S=A(X, I) + B

```

2.5 論理式

'F'では logical expression と云い、2つの算術式を関係演算子で結合したもの、論理定数、論理変数、論理関数またはこれらを論理演算子やカッコで結合したものである。

'A'では Boolean expression と云い、上にのべたものの他 if then else による条件付論理式を作ることでもある。

$P > Q$ なら $A = B$ かどうかによつて X1 に $R + S$ 又は $R - S$ を代入し、さもなければ $A < 100.0$ かどうかによつて X1 に $P * S$ 又は R / S を代入したいときは

```

IF(P.LE.Q) GO TO 1  X1:=if P>Q then
IF(A.EQ.B) GO TO 2    (if A=B then R+S
X1=R-S                else R-S) else
2 X1=R+S              if A<100.0 then
1 IF(A.LT.100.0) GO TO 3  P*S else R/S;
X1=P/S
3 X1=P*S

```

2.6 指示子

'F'では指示子は符号なし整数に限るが、'A'では名前(レベル)、スイッチ指示子及び if then else を含んだ式を認める。(ALGOL-60ではレベルとして名前の他に符号なし整数も認める)

$P > Q$ なら $A = B$ かどうかによつて $L1$ (又は 1) か $L2$ (又は 2) に分岐し、さもなければ $A < 100.0$ かどうかによつて $L3$ (又は 3) か $L4$ (又は 4) に分岐したいときは

IF(P.GT.Q) GO TO 5	go to if $P > Q$ then
IF(A.LT.100.0) GO TO 3	(if $A = B$ then $L1$
GO TO 4	else $L2$) else if
5 IF(A.EQ.B) GO TO 1	$A < 100.0$ then $L3$
GO TO 2	else $L4$;

3° 代入命令 (assignment) $\tilde{S}$

'F' では $\tilde{I} = \tilde{A}\tilde{E}$
'A' では $\tilde{I} := \tilde{A}\tilde{E}$;

で何れも右辺の型から左辺の型への変換は自動的に行なわれ右辺 real 左辺 integer のときは絶対値が $|\tilde{A}\tilde{E}|$ を越えない最も $\tilde{A}\tilde{E}$ に近い整数となる。

4° GO TO 命令 (go to $\tilde{S}$)

4.1 無条件 GOTO 命令

'F' では GO TO <label>

'A' では GO TO <designational expression>

で 'A' では label の他にスイッチ指示子、条件付指示式などが許される。

4.2 条件付 GO TO 命令

$A > B$ なら $L4$ (又は 4) さもなければ $L5$ (又は 5) に行きたいとき

IF(A.LT.B) GO TO 4	go to if $A > B$ then
GO TO 5	$L4$ else $L5$;

$A * B < 0$ なら $L1$ (又は 1) $= 0$ なら $L2$ (又は 2) > 0 なら $L3$ (又は 3) へ行きたければ

IF(A*B) 1, 2, 3	go to if $A * B < 0$ then $L1$ else
	if $= 0$ then $L2$ else $L3$;

4.3 computed GO TO 又はスイッチ

J が 1 であると共に I が 1 なら $L11$ (又は 11) 2 なら $L12$
(又は 12) 3 なら $L13$ (又は 13) へ行き、 J が 2 であると共に
 I が 1 なら $L21$ (又は 21)、2 なら $L22$ (又は 22)、3 なら
 $L23$ (又は 23) へ行きたいければ

```
GO TO (1, 2), J          switch S1:=L11,L12,L13;
1 GO TO (11, 12, 13), I  switch S2:=L21,L22,L23;
2 GO TO (21, 22, 23), I  switch S:=S1(I),S2(I);
                           go to S(J);
```

5° 条件付命令

5.1 一般

'F' では IF (論理式) $\bar{S}$
によつて真なら $\bar{S}$ を実行、偽なら次の $\bar{S}$ へ行く

'A' では if $\bar{BE}$ then $\bar{US}$;
if $\bar{BE}$ then $\bar{FS}$;
if $\bar{BE}$ then $\bar{US}$ else $\bar{S}$;

の 3 種がある。真のとき then の後ば実行される。

'A' の場合は else の後に又 if が来ても良いし、then の後も
begin end でつつむなら if が来ても良い。

5.2 算術 IF $\bar{S}$

'F' では IF ($\bar{AE}$) n_1, n_2, n_3

がある $\bar{AE} < 0$ なら n_1
 $= 0$ なら n_2
 > 0 なら n_3

にコントロールが変更される。

'A' では if $\bar{AE} < 0$ then go to n_1 else if $\bar{AE} = 0$
then go to n_2 else go to n_3 ; か又は go to n_i の代り
に実行したい $\bar{S}$ を書けばよい。これが複数個のときは begin end で
包む。

6° くり返し

'F' では DO $\bar{S}$ があり

$\overline{DO}$ <label> $\overline{I} = m_1, m_2$

又は

$\overline{DO}$ <label> $\overline{I} = m_1, m_2, m_3$

がある。 $\overline{I}$ は integer 変数で m_1 から m_3 おきに m_2 まで指定されたレベルまでの $\overline{S}$ が実行され、 m_3 が無いときは $m_3 = 1$ と解される。指定されたレベルのついた $\overline{S}$ は実行可能な $\overline{S}$ でなければならないので $GO TO \overline{S}$ や算術 $IF \overline{S}$ 、 $\overline{DO} \overline{S}$ 其他 $STOP$ 等の $\overline{S}$ は禁止されるので一般には $CONTINUE$ と書いておく。

'A' では

for $\overline{I} := < \text{for list} >$ do $\overline{S}$;

の形をとり、<for list> は

1. $\overline{AE}_1 \text{ step } \overline{AE}_2 \text{ until } \overline{AE}_3$
2. $\overline{AE}$
3. $\overline{AE} \text{ while } \overline{BE}$

4. 1~3のいくつかをカンマでつないだもの

の何れでもよいし、又 $\overline{S}$ はどの $\overline{S}$ であつてもよいし、いくつかあるときは begin end によつて1つの $\overline{S}$ に直せばよい。

又 'F' とちがつて real type のコントロール変数も自由に使用できる。 I を1から1きざみで50まで J を1から1きざみで100まで変えてマトリクス和

$C_{ij} = A_{ij} + B_{ij}$

を行うには

$\overline{DO}$ 10 $I = 1, 50$

for $I := 1 \text{ step } 1 \text{ until } 50 \text{ do}$

$\overline{DO}$ 20 $J = 1, 100$

for $J := 1 \text{ step } 1 \text{ until } 100 \text{ do}$

20 $C(I, J) = A(I, J) + B(I, J) \quad C(I, J) := A(I, J) + B(I, J);$

10 $CONTINUE$

I が1から2.5きざみで20まで5きざみで100まで $\sqrt{I}$ を計算して I と $\sqrt{I}$ をプリントするなら

$\overline{DO}$ 50 $J = 1.9$

for $I := 1 \text{ step } 2.5 \text{ until } 20,$

$AI = \text{FLOATP}(I) * 2.5 - 2.5 \quad 25 \text{ step } 5 \text{ until } 100 \text{ do}$

```

      SAI=SQRTF(AI)      PRINT('2R',I,SQRT(I));
      WRITE 100,AI,SAI
100  FORMAT(IH,2F15.7)
      50 CONTINUE
      DO 60 J=20,100,5
      SAI=SQRTF(FLOATF(J))
      WRITE 110,J,SAI
110  FORMAT(IH,13,F15.7)
      60 CONTINUE

```

7° コメント

'F'では左欄にCと書けば72桁以下何を書いても良いし'A'では
begin 及び;の次は何を書いてもよく、;で終る。又 end の後は end
else ; 以外は何を書いても良い。

8° 配列の処理

ベクトル マトリクス等は配列として扱うことができるが'F'では3次
元までに限り、'A'には制限はない。

3行5列の行列を行なりに読んでこれを転置した後、新しい行列を行ご
とに改行して印刷するプログラムを作ってみる。ただし各要素は4ケタ以
下の整数とする。

<pre> begin integer I,J; DIMENSION IA(3,5) READ 10,(IA(I,J)J= 1,5,I=1,3) 10 FORMAT(I4) DO 30 I=1,3 DO 30 J=1,5 IB=IA(I,J) IA(I,J)=IA(J,I) 30 IA(J,I)=IB WRITE 50,.(IA(I,J) J=1,3,I=1,5) </pre>	<pre> integer array IA[1:3,1:5]; READ (A); for J:=1 step 1 until 5 do begin for I:=1 step 1 until 3 do begin integer A; A:=IA[I,J]; IA[I,J]:=IA[J,I]; IA[J,I]:=A; PRINT('I',A); end; end; </pre>
--	--

```
50 FORMAT(4I4/)
```

```
STOP
```

```
PRINT('/');
```

```
end;
```

```
end;
```

N個の浮動小数点モードの情報をNを先頭に読み、その逆数の和を印刷せよ、ただし0の逆数は0とする($N \leq 100$)

<pre> DIMENSION A(100),B(100) READ 10,N 10 FORMAT(I3) READ 20,(A(I),I=1,N) 20 FORMAT(F15.7) SUM=0.0 DO 30 I=1,N IF(A(I))1,2,1 2 B(I)=0.0 GO TO 30 1 B(I)=1.0/A(I) 30 SUM=SUM+B(I) WRITE 40,SUM 40 FORMAT(F15.7) STOP </pre>	<pre> begin integer N,I; real SUM; array A[1:100]; READ(N); READARRAY(A,1,N); SUM:=0; for I:=1 step 1 until N do SUM:=SUM+(if A[I]=0 then 0 else 1/A [I]); PRINT('R/',SUM); end; </pre>
---	---

$y = x^5 + 4x^3 + 3$ について $x = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$ に対して計算し、各値及び最大値をMAXY=として印刷せよ

<pre> DIMENSION IY(11) DO 40 I=1,11 IX=I-6 IY[I]=X**5+4*X **3+3; </pre>	<pre> begin integer X,Y,MAXY; array YA[-5:5]; for X:=-5 step 1 until 5 do begin Y:=X↑5+4* </pre>
---	--

電算 運営資料

```

      IF(I-1)10,20,10
10  IMAX=IY(I)
      GO TO 40
20  IF(IMAX-IY(I))30,40,40
30  IMAX=IY(I)
40  CONTINUE
      WRITE 50,IMAX
50  FORMAT(I10)
      STOP

```

```

      X↑3+3;
      YA(X):=Y;
      end;
      MAX:=YA(-5);
      for X:=-5 step 1 until 3 do
      begin
        if MAX<YA(X) then
          MAX:=YA(X);
        PRINT('I',YA(X));
      end;
      PRINT('/' MAXY='I/',MAX);
      end;

```

9° スイッチ

条件によつて分岐するために 'F' には computed GO TO S が、
'A' にはスイッチがある。

クリスマスカードを送るのに相手がフランス語を解するなら 1、日本語
を解するなら 2、英語を解するなら 3 を入れて、且つ受取つた日が 12 月
1 日から 25 日までならそれぞれ

JOYEUX NOEL

クリスマス おめでとう

MERRY CHRISTMAS

12 月 26 日より 1 月 1 杯はそれぞれ

BONNE ANNEE

新年おめでとう

A HAPPY NEW YEAR

とし 2 月から 11 月までは白紙にして印刷するプログラムは

```
LOGICAL I, J
READ 10, N, D
10  FORMAT(11, F4.2)
    J=D.GT.2.1.OR.D.LT.11.30
    I=J.AND.D.GE.12.01.AND.D.LE.12.25
    GO TO (20, 30, 40), N
20  IF(I) WRITE 201
201  FORMAT(11H JOYEUXb NOEL)
    IF(J.AND..NOT.I) WRITE 202
202  FORMAT(11H BONNEb ANNEE)
30  IF(I) WRITE 301
301  FORMAT(18H KURISUMASUb OMEDETO)
    IF(J.AND..NOT.I) WRITE 302
302  FORMAT(14H SINNENb OMEDETO)
40  IF(I) WRITE 401
401  FORMAT(15H MERRYb (HRISTMAS)
    IF(J.AND..NOT.I) WRITE 402
402  FORMAT(16H ABHAPPYb NEWb YEAR)
    WRITE 500
500  FORMAT(1)
    STOP
```

電算通盤資料

```
begin      integer N,I;
           real D;
           switch SF:=FC,FN;
           switch SN:=NC,NN;
           switch SE:=EC,EN;
           switch S:=SF[I]
             SN[I],SE[I];
           procedure G; go to END;
           FC:PRINT("JOYEUX#NOEL");G;
           FN:PRINT("BONNE#ANNEE");G;
           NC:PRINT("KURISUMASU#OMEDETO");G;
           NN:PRINT("SINNEN#OMEDETO");G;
           EC:PRINT("MERRY#CHRISTMAS");G;
           EN:PRINT("A#HAPPY#YEAR");G;
           READ(N,D);
           I:=if D>2.1∨D<11.30
              then (if D ≥ 12.01 ∧
                   D ≤ 12.25 then 1 else 2)
              else 2;
           go to S[N];
           END:PRINT ('/');
end;
```

10° 'SUBROUTINE' 又は procedure

まとまつた手続をある名前とパラメタ(ない場合もある)で呼び出して
 使用するために 'F' には SUBROUTINE, 'A' には procedure がある。

'F' では dummy argument として添字なし変数が配列の名前が他の
 Function, subroutine の名前が許されるし、'A' では formal
 parameter として変数、配列、他の procedure 他の function
 designator, label, switch, string が許される。

'F' ではこの使用は CALL statement により、'A' では procedure
 statement によつて行なう。

$\sum x + \sum x^2 + \sum x^3$ を x を読んで計算し、結果をプリントさせるプログラ
 ムを作つてみる。データは個数 N を先頭に x が N 個並んでいるものとする。
 ($N \leq 100$)

<pre> DIMENSIONE(100) READ 10,N 10 FORMAT(I3) READ(A(I),I=1,N) CALL SUM(N,1,A,SX) CALL SUM(N,2,A,SXX) CALL SUM(N,3,A,SXXX) WRITE 20,SX+SXX+SXXX 20 FORMAT(I81) STOP SUBROUTINE SUM(N,P,AS) S=0 DO 30 I=1,N 30 S=S+A(I)**P RETURN END </pre>	<pre> begin real X,SX,SXX,SXXX; integer N; array, A[1:100]; procedure SUM(N,P,A,S); value N,P; integer N,P; array A; real S; begin integer I; S:=0; for I:=1 step 1 until N do S:=S+A(I)[↑]P; end; READ(N); READ ARRAY(A,1,N); SUM(N,1,A,SX); SUM(N,2,A,SXX); SUM(N,3,A,SXXX); PRINT('R1',SX+SXX+SXXX); end; </pre>
---	--

11° FUNCTION 又は function designator

10 章と殆んど同じであるが 1 価関数の場合自分自身の名前に答を代入し、式の中で直接使用することができるものとして 'F' に FUNCTION, 'A' に function designator がある。

10 章の例を書き直すと

<pre> DIMENSION A(100) READ 10, N 10 FORMAT(I3) READ(A(I), I=1, N) WRITE 20, SUM(N, 1, A) + SUM(N, 2, A) + SUM(N, 3, A) 20 FORMAT(I8/) STOP FUNCTION SUM(N, P, A) SUM=0 DO 30 I=1, N 30 S=S+A(I)**P RETURN END </pre>	<pre> begin real X; integer N; array A {1:100}; real procedure SUM(N, P, A); value N, P; integer N, P; array A; begin integer I; SUM:=0; for I:=1 step 1 until N do SUM:=SUM + A(I)↑P; end; end; READ(N); READARRAY(A, 1, N); PRINT('R/', SUM(N, 1, A) +SUM(N, 2, A)+SUM(N, 3, A)); end; </pre>
---	---

12° 標準関数等

12.1 標準関数

宣言を要しないで使用できる標準関数として

'F' には

組込み関数	ABSF (real)
	XABSF (integer)
ライブラリ関数	LOGF (real)
	SINF (real)
	COSF (real)
	EXPF (real)
	SQRTF (real)
	ATANF (real)
	TANHF (real)
	MODF (real) 'A' の entier に当る
	INTF (real) 切りすて
	FLOATF (integer)
	XFIXF (real)
	SIGNF (A_1, A_2) A_1 の絶対値の A_2 の符号

等があり

'A' には ABS, LN, EXP, SIN, COS, ARCTAN, SQRT, SIGN, FLOAT, FIX がある。(ALGOL 60 では小文字で FLOAT, FIX がなく entier がある)

SIGN は負なら -1, 0 なら 0, 正なら +1 となる。

12.2 演算ステートメント関数

'F' には $TAN(X) = SIN(X) / COS(X)$

という形で演算ステートメント関数が使用できる。

'A' では関数指示子宣言を行つて

```
real procedure TAN(X)
  value X; real X;
  TAN := SIN(X) / COS(X);
```

としておいて本文で $TAN(X)$ を中の中 で用いることになる。

12.3 EXTERNAL

'F' には EXTERNAL $\tilde{I}_1, \dots, \tilde{I}_n$ 'A' には external $\tilde{I}_1, \dots, \tilde{I}_n$; 又は external <type> $\tilde{I}_1, \dots, \tilde{I}_n$; がある。すでに用意してある手続の宣言である。

お わ り に

以上個別に比較したものを本質的な相違点のみとり出して比較してみると次のようなものになろう。

問 題 点	'F'	'A'
1. ブロック構造	ない。従つて特に共通な名前の定義のために <i>COMMON</i> ，部分的に番地を共有するために <i>EQUIVALENCE</i> を用い面倒になる代りに無宣言で名前を使用できる。(但し <i>I</i> ~ <i>K</i> が先頭にくると <i>integer</i> 扱いになるので注意を要する)	ある。従つて部分的に使いすてるものはブロックの中で宣言すればよい。その代りすべての名前には宣言を必要とする。
2. <i>integer real</i> の混合演算	認めない (換算してから計算)	認める
3. 配列の次元数に関する宣言	'F II' では 2 次元 'F IV' では 3 次元まで	無制限
4. くり返しのコントロール変数 (又は <i>index</i>)	<i>integer</i> に限る (<i>DO</i> の後のステートメント中で適当に変換する必要がある)	制限なし
5. 倍精度演算	<i>DOUBLE</i> 宣言でよい (但し 'F IV')	<i>B5500</i> の <i>DOUBLE</i> 可変長語の <i>FACOM</i> などを除き、特別な措置を要する。

6. 複素数宣言	COMPLEX 宣言でよい (但し 'FN')	procedure を作る必要がある。
7. 条件式	ない、条件命令をいちいち出して行なう要あり IF (C=D) 10 IF (C.GT.10) 20 IF (E.NE.F) 30 B=A(S,P,Q,V)+A2*C2 GO TO 80 30 B=A(S,P,Q,V)+A1*C2 GO TO 80 20 IF (E.NE.F) 40 B=A(S,P,Q,U)+A2*C2 GO TO 80 40 B=A(S,P,Q,U)+A1*C2 GO TO 80 10 IF (C.GT.10) 50 IF (E.NE.F) 60 B=A(R,P,Q,V)+A2*C2 GO TO 80 60 B=A(R,P,Q,V)+A1*C2 GO TO 80 50 IF (E.NE.F) 70 B=A(R,P,Q,U)+A2*C2 GO TO 80 70 B=A(R,P,Q,U)+A1*C2 80	算術式、論理式、指示式に対してある。 B:=A{ if C=D then R else S, P,Q, if C>10 then U else V} +(if E≠F then A1 else A2) * C2!

であるがしかも 'F' では4次元の配列
 $A(R, P, Q, V)$ は不可能である。

電算運営資料

従つて以上を要するに4次元以上の配列が必要となるときはそれを問題の方で避ける様に工夫し、real integer の混合演算に気をつけ、何人かで共同作業をするときに共通の名前に気をくばり、条件式が出てくるときはこれを多少面倒でも条件命令に分解する労をいとわないならば、プログラミングだけを専門とする者にとっては 'F' も 'A' も同じことであろう。しかし仕事の途中で自分のプログラムを書く積りの人にとっては、何が何でも理屈なしに例外的な規則を棒暗記することが苦にならない人は別として、スジの通つた文法によつて自由に我々の考えたままの道筋に従つてプログラムを書き下して行こうとする者にとっては 'A' が向いているといふことができる。

§ 2 標準手続プログラム紹介

(1) 標準手続および一般手続の一覧表とその内容について

(その2)

EXTERNAL PROCEDURE

登録番号	手続の内容および標題	プログラマー
EX1035	連立一次方程式 (Gauss基本法)	LEGAB 沖 電 気
EX1036	連立一次方程式 (Gauss sweep out法)	GSWPOUT "
EX1037	連立一次方程式 (Gauss Seidel法)	GSEIDEL "
EX1038	ベクトル和	VECSUM "
EX1039	行列和	MATSUM "
EX1040	ベクトル差	VECSUB "
EX1041	行列差	MATSUB "
EX1042	ベクトル内積	INNER PRODUCT "
EX1043	行列積	MAT PRODUCT "
EX1044	転置行列	TRANSPOSED MATRIX "
EX1045	逆行列	INVERSE MATRIX "
EX1046	行列式の値	DETERMINANT "
EX1047	$\sin hx$	HSIN "
EX1048	$\cos hx$	HCOS "
EX1049	算術平均値、分散、標準偏差	TOKEI KEISAN "
EX1050	相関係数 (1)	SOUKAN "
EX1051	相関係数 (2)	SOKAN "
EX1052	回帰係数	KAIKI "
EX1053	最小二乗近似 (1)	LESQU1 "
EX1054	最小二乗近似 (2)	LESQU2 "
EX1055	チェブシェフ近似	CHEBYSHEV "
EX1056	三項方程式	THREE "
EX1057	積分 (SIMPSON法、一変数)	SIMPSON "
EX1058	誤差関数	ERROR FUNCTION "
EX1059	$\sin^{-1}x$	ASIN "
EX1060	$\cos^{-1}x$	ACOS "
EX1061		
登録番号	手続の内容および標題	プログラマー
PP1015	アーチダム型棒計算 その1	飯田隆一
PP1016	" " " " その2	"
PP1017	模型実験応力解析	"
PP1018	一元配置の分散分析を行なう手続	AOV.1 中村慶一

EXTERNAL PROCEDURE

プロシデア名		LEGAB(PN, PA, PX, X1, ZER).....連立一次方程式		
登録番号		EX1035	製作者	沖電気
概 説	概	〈目的〉 連立一次方程式の根をGaussの基本法で求める。 〈パラメータ〉 PN:元数(integer) PA: arrayの名前(veal)PA(1:n, 1:m)でnは元数、m=n+1である。係数及び定数項をn行m列の行列として格納する。 PX: arrayの名前、(n)PX(1:n)でnは元数である。実行後に根を格納する。 X1: arrayの名前、(n)X1(1:m)でm=n+1とする。Working Storageとして使う。 ZER: 相対的ゼロ十分小さい数値を格納する(real) 〈制限〉 元数の制限は原則としてないが、75元までは通った例がある。 〈使用法〉 形式パラメータPAに対する実パラメータに係数および定数項をPNに元数を格納して呼び出せばPXに添数の小さい方から順に解を得る。 〈注意〉 形式パラメータPN、ZERはvalue指定してある。 実行中Pivotの絶対値がゼロでかつ同列の下位要素の絶対値がすべてゼロのとき[DATA CHECK]と印字して実行を中止してメインに戻る。また実行後PAの内容はすべてこわれる。		
	説			
プロシデア名		GSWPOUT(PN, PA, PB, PX, ZER).....連立一次方程式		
登録番号		EX1036	製作者	沖電気
概 説	概	〈目的〉 連立一次方程式の根をGauss sweep out法で求める。 〈パラメータ〉 PN:元数(integer) PA: Gauss 基本法に同じ PB: Gauss 基本法のX1に同じ PX: Gauss 基本法に同じ ZER: " " 〈制限〉 Gauss 基本法に同じ 〈使用法〉 " 〈注意〉 形式パラメータPN、ZERはvalue指定してある。		
	説			

EXTERNAL PROCEDURE

プロセデュア名	GSEIDEL (PN, PEPS, PA, PX)..... 連立一次方程式		
登録番号	EX1037	製作者	沖 電 気
概	〈目 的〉 連立一次方程式を Gauss Seidel 法で求める。 〈パラメータ〉 PN: 元数 (integer) PEPS: 収束判定値 (real) PA: array の名前 (real) PA(1:n, 1:m) で n は元数, m=n+1 である。係数及定数項 n 行 m 列の行列として格納する。 PX: array の名前 (real), PX(1:n) で実行後に根の格納を行なう。 〈制 限〉 Gauss 基本法に同じ 〈使用法〉 " " 〈注 意〉 形式パラメータ PN, PEPS は value 指定してある。 与えられた PA に於て各行の要素をその対角要素で割りその後で対角要素以外の要素の絶対値の和が 1 より大きい場合 [DATA CHECK] と印字してメインに帰る。尚この場合対角要素がゼロか否かはチェックしていない。 実行後 PA の内容はすべてこわれる。また与えられた収束判定値に達するまで何回でも反復する。		
説			
プロセデュア名	VECSUM (N, A, B, C)..... ベクトル和		
登録番号	EX1038	製作者	沖 電 気
概	〈目 的〉 数ベクトル $a=(a_1, a_2, \dots, a_n)$, $b=(b_1, b_2, \dots, b_n)$ をともに n 次元とするととき $a+b=(a_1+b_1, a_2+b_2, \dots, a_n+b_n)$ を求める。 〈パラメータ〉 N: 次元 (integer) A: $a=(a_1, a_2, \dots, a_n)$ の array の名前 (real) B: $b=(b_1, b_2, \dots, b_n)$ の array の名前 (real) C: $a+b=(a_1+b_1, a_2+b_2, \dots, a_n+b_n)$ の求まる array の名前 (real) 〈制 限〉 形式パラメータ A, B, C に対応する配列は 1 次元でその添数の下限は 1 で上限は N までである。 〈使用法〉 形式パラメータ N, A, B に対応する実パラメータに値を与えて呼べばパラメータ C にベクトルの和が求まる。		
説	〈注 意〉 形式パラメータ N は value 指定してある。		

EXTERNAL PROCEDURE

プロセデュア名		MATSUM(M, N, A, B, C)..... 行列和		
登録番号		EX1039	製作者	沖 電 気
概 説	《目 的》	行列 $A = (a_{ik})$ 、 $B = (b_{ik})$ の和 $A+B = (a_{ik}+b_{ik})$ を求める。		
	《パラメータ》	M: 行数 (integer) N: 列数 (integer) A: 行列 $A = (a_{ik})$ のarrayの名前 (real) B: 行列 $B = (b_{ik})$ のarrayの名前 (real) C: 行列和 $A+B$ の求まるarrayの名前 (real)		
	《制 限》	形式パラメータA、B、Cに対応するarrayは2次元でその添数の下限はいずれも1で上限はMおよびNまでである。		
	《使用法》	形式パラメータM、N、A、Bに対応する実パラメータに値を与えて呼べば形式パラメータCに対応する実パラメータに行列和が求まる。		
	《注 意》	形式パラメータはvalue指定してある。		
プロセデュア名		VECSUB(N, A, B, C)..... ベクトル差		
登録番号		EX1040	製作者	沖 電 気
概 説	《目 的》	数ベクトル $a = (a_1, a_2, \dots, a_n)$ 、 $b = (b_1, b_2, \dots, b_n)$ をともにn次元とするととき差 $a-b = (a_1-b_1, a_2-b_2, \dots, a_n-b_n)$ を求める。		
	《パラメータ》	N: 次元 (integer) A: $a = (a_1, a_2, \dots, a_n)$ のarrayの名前 (real) B: $b = (b_1, b_2, \dots, b_n)$ のarrayの名前 (real) C: $a-b = (a_1-b_1, a_2-b_2, \dots, a_n-b_n)$ の求まるarrayの名前 (real)		
	《制 限》	形式パラメータA、B、Cに対応するarrayは1次元でその添数の下限は1で上限はNまでである。		
	《使用法》	形式パラメータN、A、Bに対応する実パラメータに値を与えて呼べば形式パラメータCに対応する実パラメータにベクトルの差が求まる。		
	《注 意》	形式パラメータNはvalue指定してある。		

EXTERNAL PROCEDURE

プロセデュア名		MATSUB(M, N, A, B, C).....行列差	
登録番号		EX 1041	製作者 沖 電 気
概 説	《目的》	行列 $A=(a_{ik})$, $B=(b_{ik})$ の差 $A-B=(a_{ik}-b_{ik})$ を求める。	
	《パラメータ》	M: 行数 (integer) N: 列数 (integer) A: 行列 $A=(a_{ik})$ のarrayの名称 (real) B: 行列 $B=(b_{ik})$ のarrayの名称 (real) C: 行列差 $A-B$ の求まる arrayの名称 (real)	
	《制限》	形式パラメータA, B, Cに対応するarrayは2次元で、その添数の下限はいずれも1で上限はMおよびNまでである。	
	《使用法》	形式パラメータM, N, A, Bに対応する実パラメータに値を与えて呼べば形式パラメータCに対応する実パラメータに行列差が求まる。	
	《注意》	形式パラメータM, Nはvalue指定してある。	
プロセデュア名		INNERPRODUCT(N, X, Y, XY).....ベクトル内積	
登録番号		EX 1042	製作者 沖 電 気
概 説	《目的》	2個ベクトル $x=(x_1, x_2, \dots, x_n)$, $y=(y_1, y_2, \dots, y_n)$ をとともにn次元とするとき内積 $(x, y)=x_1y_1+x_2y_2+\dots+x_ny_n$ を求める。	
	《パラメータ》	N: 次元 (integer) X: $x=(x_1, x_2, \dots, x_n)$ のarrayの名称 (real) Y: $y=(y_1, y_2, \dots, y_n)$ のarrayの名称 (real) XY: 内積の求まる変数 (real)	
	《制限》	形式パラメータX, Yに対応するarrayは1次元でその添数の下限は1で上限はNまでである。	
	《使用法》	形式パラメータN, X, Yに対応する実パラメータに値を与えて呼び出せば形式パラメータXYに対応する実パラメータに内積が求まる。	
	《注意》	形式パラメータNはvalue指定してある。	

標準手続プログラム紹介

EXTERNAL PROCEDURE

プロセデュア名		MATPRODUCT (L, M, N, A, B, C).....行列積		
登録番号		EX1043	製作者	沖電気
概 説	〈目的〉	行列 $A=(a_{ij})$ と $B=(b_{ij})$ に於て、 $A(1, m)$ 型、 $B(m, n)$ 型の行列とするとき $C_{ik}=\sum_{j=1}^m a_{ij}b_{jk}$ を (i, K) 組成分子とする $(1, n)$ 型の行列(A と B の積)を求む。		
	〈パラメータ〉	L: 行列Aの行数(integer) M: 行列Aの列数および行列Bの行数(integer) N: 行列Bの列数(integer) A: 行列 $A=(a_{ij})$ のarrayの名前(real) B: 行列 $B=(b_{ij})$ のarrayの名前(real) C: 行列の積 AB の求まるarrayの名前(real)		
	〈制限〉	形式パラメータA、B、Cに対応するarrayは2次元でその添数の下限はいずれも1である。		
	〈使用法〉	形式パラメータL、M、N、A、Bに対応する実パラメータに値を与えて呼びだせば形式パラメータCに対応する実パラメータに行列積 AB が求まる。		
	〈注意〉	形式パラメータL、M、Nはvalue指定してある。 行列積 AB は行列Aの列数と行列Bの行数が一致する場合のみ定義される。		
プロセデュア名		TRANSPSED MATRIX (M, N, A, B).....転置行列		
登録番号		EX1044	製作者	沖電気
概 説	〈目的〉	(m, n) 行列Aの転置行列、 (n, m) 行列 A^t を求める。		
	〈パラメータ〉	M: 行列Aの行数(integer) N: 行列Aの列数(integer) A: 行列Aのarrayの名前(real) B: 行列Aの転置行列 A^t の求まるarrayの名前(real)		
	〈制限〉	形式パラメータA、Bに対応するarrayは2次元でその添数の下限は、いずれも1で上限はMおよびNまでである。		
	〈使用法〉	形式パラメータM、N、Aに対応する実パラメータに値を与えて呼べば形式パラメータBに対応する実パラメータに行列Aの転置行列 A^t が求まる。		
	〈注意〉	形式パラメータM、Nはvalue指定してある。		

標準手続プログラム紹介

-67-

EXTERNAL PROCEDURE

プロセデユア名		HSIN (X, Y)..... sin hx	
登 録 番 号		EX 1047	製 作 者 沖 電 気
概 説	《目 的》	sin hx を求める。	
	《パラメータ》	X:角(ラジアン)(real) Y: sin hx の求まる変数(real)	
	《制 限》	形式パラメータXに対応する実パラメータに与える値、Xの範囲は $-1 \leq X \leq 1$ である。	
	《使用法》	形式パラメータXに対応する実パラメータに角Xラジアンを与えると、形式パラメータYに対応する実パラメータに sin hxの値が求まる。	
	《注 意》	形式パラメータXはvalue指定してある。 最大誤差 2.5×10^{-11}	
	《方 法》	テイラー展開	
プロセデユア名		HCOS (X, Y)..... Cos. hx	
登 録 番 号		EX 1048	製 作 者 沖 電 気
概 説	《目 的》	cos hx を求める。	
	《パラメータ》	X:角(ラジアン)(real) Y: cos hxの求まる変数(real)	
	《制 限》	形式パラメータXに対応する実パラメータに与える値Xの範囲は $-1 \leq X \leq 1$ である。	
	《使用法》	形式パラメータXに対応する実パラメータに角Xラジアンを与えると形式パラメータYに対応する実パラメータに cos hxの値が求まる。	
	《注 意》	形式パラメータXはvalue指定してある。 最大誤差 1.1×10^{-12}	

EXTERNAL PROCEDURE

プロセデュア名		TOKEI KEISAN (A, N, MEAN, VARIAN, STDEV)		
登録番号		EX 1049	製作者	沖 電 気
概 説	《目 的》	N個のデータ $x_1, x_2, \dots, x_n$ に対してその算術平均値 $\bar{x}$ 、分散 v 及び標準偏差 SD を求める。		
	《パラメータ》	A: データの入っている array の名前 (real) N: データの個数 (integer) MEAN: 算術平均の求まる変数 (real) VARIAN: 分散の求まる変数 (real) STDEV: 標準偏差の求まる変数 (real)		
	《制 限》	形式パラメータ A に対応する実パラメータの配列の次元は 1 次元、係数の下限は 1、上限は N である。		
	《使用法》	A, N に対応する実パラメータに値を与えれば結果が MEAN, VARIAN, STDEV に求められる。		
	《注 意》	形式パラメータ N は value 指定してある。		
プロセデュア名		SOUKAN (R) 相関係数(1)		
登録番号		EX 1050	製作者	沖 電 気
概 説	《目 的》	N組の x, y のデータに対して x と y の単相関係数 R を求める。		
	《パラメータ》	R: 相関係数の求まる変数 (real)		
	《使用法》	データの穿孔形式は次のようにする。 N, $x_1, y_1, x_2, y_2, \dots, x_n, y_n$; (T) (T) N: データの組数 (integer) $x_1, y_1, \dots$; 個々のデータ (real)		

EXTERNAL PROCEDURE

プロセデュア名		SOKAN (A, N, R)..... 相関係数 (2)		
登録番号		EX1051	製作者	沖 電 気
概 説	〈目的〉	N組のx, yのデータに対して、xとyの単相関係数Rを求める。		
	〈パラメータ〉	A: データの入っているarrayの名前 (real) N: データの組数 (integer) R: 相関係数の求まる変数 (real)		
	〈制限〉	形式パラメータに対応する実パラメータの配列の次元は1次元、添数の下限は1、上限は2Nである。尚配列の順序はx ₁ , y ₁ , x ₂ , y ₂ , ..., x _N , y _N とする。		
	〈使用法〉	A, Nに対応する実パラメータに値が与えられたとき結果がRに求まる。		
	〈注意〉	形式パラメータNはvalue指定してある。		
プロセデュア名		KAIKI (A)..... 回帰係数		
登録番号		EX1052	製作者	沖 電 気
概 説	〈目的〉	x, yのN組のデータに対して回帰方程式 $y - \bar{y} = a(x - \bar{x})$ の勾配aを求める。		
	〈パラメータ〉	A: 回帰係数の求まる係数 (real)		
	〈使用法〉	データの穿孔形式は、N, x ₁ , y ₁ , x ₂ , y ₂ , ..., x _N , y _N とする。 上記データテープに対してこの手続を呼べば結果がRに求まる。		

EXTERNAL PROCEDURE

プロセデュア名		LESQU1(A, N).....最小二乗近似 (1)		
登録番号		EX1053	製作者	沖 電 気
概 説	〈目 的〉	m組のデータ(x ₁ , y ₁), (x ₂ , y ₂), ..., (x _m , y _m) に対する最小二乗近似多項式 $y = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n$ の係数 a ₀ , a ₁ , a ₂ , ..., a _n を求める。		
	〈パラメータ〉	A: 係数 a _n の配列の名前 (real) N: 近似多項式の最高次数 n (integer)		
説	〈制 限〉	形式パラメータ A に対応する実パラメータの配列は 1 次元でその係数の下限は 0、上限は N である。但し $N \leq 30$ とする。		
	〈使用法〉	データテープの穿孔形式は m, x ₁ , y ₁ , x ₂ , y ₂ , ..., x _n , y _n ; とし、m は integer、個々のデータは real 形とする。		
	〈注 意〉	このデータを用いてこの手順を呼べば A に対応する実パラメータに結果が得られる。		
	〈注 意〉	形式パラメータ N は value 指定してある。 m < n のときは MISUSE と印字しメインプログラムに戻る。		
プロセデュア名		LESQU2(A, N, M, X, Y, GOSA).....最小二乗近似 (2)		
登録番号		EX1054	製作者	沖 電 気
概 説	〈目 的〉	LESQU1 に同じ		
	〈パラメータ〉	A: 係数 a _n の配列の名前 (real) N: 近似多項式の最高次数 (integer) M: x, y の組数 (integer) X: データ x _i (i=1, 2, ..., m) の配列の名前 (real) Y: データ y _i (i=1, 2, ..., n) の配列の名前 (real) GOSA: 最小二乗和 (real)		
説	〈制 限〉	形式パラメータ A は A(0:N) の形である。ただし $N \leq 30$ である。 形式パラメータ X, Y は X(1:M), Y(1:M) の形である。		
	〈使用法〉	形式パラメータ N, M, X, Y に対応する実パラメータに値が与えられたとき、呼び出せば A に対応する実パラメータに係数、GOSA に対応する実パラメータに誤差二乗和が求められる。		
	〈注 意〉	形式パラメータ M, N は value 指定してある。 N が 30 を越えた場合 → OVER NO CALCULATION } と印字しメインに戻る。 M < N の場合 → MISUSE }		

EXTERNAL PROCEDURE

プロセデュア名		CHEBYSHEV (A, B, FUN, PP, N)..... チェブシエフ近似		
登録番号		EX1055	製作者	沖 電 気
概 説	《目 的》	一関数 $f(x)$ の区間 $a \leq x \leq b$ に於けるチェブシエフ近似多項式 $y = P_0 + P_1 x + P_2 x^2 + \dots + P_n x^n$ の係数 $P_0, P_1, P_2, \dots, P_n$ を求める。		
	《パラメータ》	A : 区間の下限 (real) B : 区間の上限 (real) FUN : 関数 $f(x)$ の手続きの名前 PP : 係数 $P_i (i=0, 1, 2, \dots, n)$ の配列の名前 (real) N : 近似式の最高次数		
説	《制 限》	形式パラメータ PP は $PP(O:N)$ の形である。但し $0 \leq N \leq 20$ とする。 形式パラメータ FUN に対応した実パラメータの手続きはパラメータのない手続きでその本体で自変数 XX という non local な real 形単純変数、従属変数は FF という non local な real 形単純変数でなければならない。		
	《使用法》	形式パラメータ A, B, N に対応する実パラメータに値が与えられたとき結果が PP に対応する実パラメータに得られる。		
	《注 意》	形式パラメータ A, B, N は value 指定してある。N が 20 をこえた指定をした時は N OVER NO. CALCULATION と印字してメインプログラムに戻る。		
プロセデュア名		THREE (PN, PA, PCOF, PX, ZER)..... 三項方程式		
登録番号		EX1056	製作者	沖 電 気
概 説	《目 的》	n 元三項方程式の根を求める。		
	《パラメータ》	PN : 元数 (real) PA : 配列の名前 (real) $PA(1:n, 1:m)$ で n は元数、 $m = n+1$ である。 Working Storage として使う。 PCOF : 配列の名前 (real) $PCOF(1:k)$ で $k = 4n-2$ である。 係数及び定数項をベクトルの形に格納すること。 PX : 配列の名前 (real) $PX(1:n)$ である。実行後に根 $(x_1, \dots, x_n)$ が格納される。		
説	《制 限》	ZER : 相対的ゼロ (real) 十分小さい値を与えること。 元数に対する制限は原則としてないが記憶容量より考えること。		
	《使用法》	形式パラメータ PCOF に対する実パラメータに係数及び定数を同 PN に対するそれに元数を格納すれば実行後形式パラメータ PX に対応する実パラメータに根が求まる。		
	《注 意》	形式パラメータ PN, ZER は value 指定してある。 Pivot の絶対値が ZER より小の場合は [DATA CHECK] と印字してメインプログラムに戻る。 実行後 PCOF の内容は保存されている。		

EXTERNAL PROCEDURE

標準手続プログラム 紹介

プロセデュア名		SIMPSON (AA, BB, EE, SIMP, FUNC)..... 積分		
登録番号		EX1057	製作者	沖 電 気
概 説	《目 的》	y = f(x)において区間 (a, b) の y の積分値を指定する精度までシン プソン法で求める。		
	《パラメータ》	AA : 求める区間の下限 (real) BB : 求める区間の上限 (real) EE : 収束判定のための値 (real) SIMP : 積分値の求まる変数 (real) FUNC : 関数 f(x) の手続きの名前		
	《制 限》	手続き FUNC は、パラメータのない手続きでその本体に於て自変数は XX という real 形単純変数、従属変数は FF という real 形単純変数で あらわされていなければならない。		
	《使用法》	AA, BB, EE に対する実パラメータに値が与えられているとき実行 後 SIMP に対応する実パラメータに結果が得られる。		
	《注 意》	形式パラメータ AA, BB, EE は value 指定してある。		
プロセデュア名		ERROR FUNCTION (F, X)..... 誤差関数		
登録番号		EX1058	製作者	沖 電 気
概 説	《目 的》	$\text{erf } x = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ を求める。		
	《パラメータ》	F : 誤差関数値の求まる変数 (real) X : x の値 (real)		
	《制 限》	形式パラメータ X に対応する実パラメータに与える値 x の範囲は $0 \leq x \leq 10$ である。		
	《使用法》	形式パラメータ X に対応する実パラメータに x を与えて呼べば形式パラ メータ F に対応する実パラメータに結果が得られる。		
	《注 意》	形式パラメータ X は value 指定してある。		

EXTERNAL PROCEDURE

プロセデユア名		ASIN(X, Y)..... $\sin^{-1}x$		
登録番号		EX1059	製作者	沖 電 気
概 説	《目的》	$\sin^{-1}x$ を求める。		
	《パラメータ》	X:角(ラジアン)(real) Y: $\sin^{-1}x$ が求まる。(real)		
	《制限》	形式パラメータXに対応する実パラメータに与える値xの範囲は $-1 \leq x \leq 1$ である。		
	《使用法》	形式パラメータXに対応する実パラメータに角xラジアンを与えると形式パラメータYに対応する実パラメータに $\sin^{-1}x$ の主値を得る。		
	《注意》	形式パラメータXはvalue指定してある。		
プロセデユア名		ACOS(X, Y)..... $\cos^{-1}x$		
登録番号		EX1060	製作者	沖 電 気
概 説	《目的》	$\cos^{-1}x$ を求める。		
	《パラメータ》	X:角(ラジアン)(real) Y: $\cos^{-1}x$ が求まる(real)		
	《制限》	形式パラメータXに対応する実パラメータに与える値xの範囲は $-1 \leq x \leq 1$ である。		
	《使用法》	形式パラメータXに対応する実パラメータに角xラジアンを与えると形式パラメータYに対応する実パラメータに $\cos^{-1}x$ の主値を得る。		
	《注意》	形式パラメータXはvalue指定してある。		

POPULAR PROCEDURE

プロセデュア名		アーチダムの型枠計算 その1		
登録番号		PP1013	製作者	ダム構造研究室長 飯田 隆一
概説	〈目的〉	アーチダムの型枠の座標規準点からの測角および各ブロック各リフトの 体積、リフト面積の計算を行う（規準点が固定されている時の三心不等 厚アーチ）		
	〈パラメータ〉	H：リフト高 D：表面から計算点までの距離 N1：ブロック数（＜40） N2：型枠上の点の数（＜16） N3：ダムの型枠式の組数（＜10） N4：規準点の組数（＜10） EL3：計算開始標高 EL4：計算終了標高 A：arrayの名前、三角点の座標 SC：arrayの名前 天端アーチの中心線にそつてのクラウンから 縫目までの弧長 B1：arrayの名前 クラウンの形状 B2：arrayの名前 アバットメントの形状 EL1：arrayの名前 型枠式に対する標高 EL2：arrayの名前 規準点に対する標高 C：arrayの名前 規準点の座標		
		〈注 意〉 なし		
プロセデュア名		アーチダムの型枠計算 その2		
登録番号		PP1014	製作者	飯田 隆一
概説	〈目的〉	アーチダムの型枠の座標、規準点からの測角および各ブロック、各リフ トの体積、リフト面積の計算を行う。（規準点が固定されていない時の 三心不等厚アーチ）		
	〈パラメータ〉	H：リフト高 D：表面からの計算点までの距離 N1：ブロック数（＜40） N2：型枠上の点の数（＜16） N3：ダムの型枠式の組数（＜10） EL3：計算開始標高 EL4：計算終了標高 SC：arrayの名前 天端アーチの中心線にそつてのクラウンから 各縫目までの弧長 A：arrayの名前 三角点の座標 B1：arrayの名前 クラウンの形状 B2：arrayの名前 アバットメントの形状 EL1：arrayの名前 型枠式に対する標高		
		〈注 意〉 なし		

標準手続プログラム紹介
POPULAR PROCEDURE

プロセデュア名		模型実験応力解析		
登録番号		PP1015	製作者	飯田 隆一
概説	〈目的〉	模型実験において測定された歪から、応力、断面力を求める。		
	〈パラメータ〉	Q: 測定点数 R: 断面力を計算するか否かの記号 E: 弾性係数 NU: ポアソン比 N: 縮尺 G: 荷重の比重 A1: arrayの名前 ゲージの符号と測定歪 A2: arrayの名前 断面係数		
説	〈使用法〉			
	〈注意〉	なし		
プロセデュア名				
登録番号			製作者	
概説				
説				

-77-

標準手続プログラム紹介

(2) プログラム紹介

一元配置の分散分析の手続

技術管理室 中 村 慶 一

1. 目 的 一元配置の分散分析を行ないたいとき、水準数をM、各水準中のくり返し数を IA_i としてデータをマトリクスXAに各行に各水準のデータが入っているように(くりかえし数が異なるときは最大のくり返し数がおさまる様に列の数を定めておけばデータの無い所は何が入つていても差し支えない)格納したものを受け取つて、

$$MX = \sum_{j=1}^{IA_i} X_{ij} / IA_i$$

$$RV = \sqrt{\sum_{j=1}^{IA_i} (X_{ij} - \bar{X}_i)^2 / IA_i}$$

を各水準ごとに計算し大きさMのベクトルに格納すると同時にMEAN RV N と打つた下に平均、不偏分散の平方根及び水準ナンバーを印刷し、次に

$$\text{全 変 動} \quad SSO = \sum_i \sum_j X_{ij}^2 - (\sum_i \sum_j X_{ij})^2 / \sum_i IA_i$$

$$\text{A 間 変 動} \quad SSA = (\sum_j X_{ij})^2 / IA_i - (\sum_i \sum_j X_{ij})^2 / \sum_i IA_i$$

$$SSE = SSO - SSA$$

$$\text{全 自 由 度} \quad NO = \sum_i IA_i - 1$$

$$\text{A 間 自 由 度} \quad NA = M - 1$$

$$\text{誤 差 自 由 度} \quad NE = NO - NA$$

$$\text{A 間 不 偏 分 散} \quad FA = SSA / NA$$

$$\text{誤 差 不 偏 分 散} \quad FE = SSE / NE$$

$$\text{分 散 比} \quad FO = FA / FE$$

を計算し

AOV TABLE

SV	SS	DF	MS	F
A	SSAの内容	NAの内容	FAの内容	FOの内容
E	SSEの内容	NEの内容	FEの内容	
O	SSOの内容	NOの内容		

という形でプリントすると同時にMX、RV、FOに結果を格納する手続きである。

なおプログラムはNEW-ALGOL-Hで書いてある。

```

procedure AOV1(M, IA, XA, MX, RV, FO);
  comment ONE WAY ANALYSIS OF VARIANCE;
  M: number of levels ( $M \leq 10$ ),
  IA: data size vector of each level (1 dimensional
      array),
  XA: data array (I: column number, J: data number) (2
      dimensional array of  $M \times \text{MAX of IA}$ ),
  MX: mean value vector (1 dimensional array),
  RV: root mean square vector (1 dimensional array),
  FO: variance ratio;
value M;
integer M; integer array IA;
array XA, MX, RV; real FO;
begin
  integer I, J, NO, NA, NE;
  real SSA, SSE, SSO, CF, FA, FO, SO, FE;
  array SX, SSX[1:10];
  PRINT(" #####MEAN#####RV#####N' / ");
  for I:=1 step 1 until M do
    begin
      real S, SS;

```

標準手続プログラム紹介

```

      S:=SS:=0;
      for J:=1 step 1 until IA(I) do
      begin
        S:=S+XA(I,J);
        SS:=SS+XA(I,J)2;
      end;
      MX(I):=S/FLOAT(IA(I));
      SSX(I):=SS;
      SX(I):=S;
      RV(I):=SQRT((SS-S2/FLOAT(IA(I)))/(FLOAT(IA(I))-1)));
      PRINT( 2(B-5Z,5DB),BZD/, MX(I),RV(I),I);
    end;
  PRINT( 3/ );SO:=SSO:=SSA:=0;NO:=NA:=0;
  for I:=1 step 1 until M do
  begin
    SSO:=SSO+SSX(I);
    SSA:=SSA+SX(I)2/FLOAT(IA(I));
    NO:=NO+IA(I);
    SO:=SO+SX(I);
  end;
  NO:=NO-1;
  NA:=M-1;
  NE:=NO-NA;
  CF:=SO2/FLOAT(NO+1);
  SSO:=SSO-CF;
  SSA:=SSA-CF;
  SSE:=SSO-SSA;
  FA:=SSA/FLOAT(NA);
  FE:=SSE/FLOAT(NE);
  FO:=FA/FE;

```

```

PRINT("#####AOV#TABLE'/'");
PRINT("##SV#####SS#####DF#####MS#####F'/'");
PRINT("##A' B-4Z.4DB,B-2ZDB,B-3Z.3DB/' ,SSA,NA,FA);
PRINT("##E' B-4Z.4DB,B-2ZDB,B-3Z.3DB/' ,SSE,NE,FE);
PRINT("-----'/'");
PRINT("##O' B-4Z.4DB,B-2ZDB/' ,SSO,NO);
end;

```

F 分布表 5 % 点の近似値を求める手続

中 村 慶 一

分散分析などで2組の分散 V_1 および V_2 の比が(ただし, 大きい方を分子にとる)ある限界より大きくなつたら分散の一樣性があるとは認められないとする危険率5%の点は分子の自由度 ϕ_1 および分子の自由度 ϕ_2 の関数でありF分布表として求められているが, 自動計算機の中で検定をやらせたいときはこの表全部を記憶させるのも大変なので近似式を記憶させてこれを用いる。表の一部を摘記すれば

$\phi_1 \backslash \phi_2$	1	3	5	10	20	60
5	6.61	5.41	5.05	4.74	4.62	4.43
10	4.95	3.71	3.33	2.98	2.85	2.62
20	4.35	3.10	2.71	2.35	2.12	1.95
120	3.92	2.68	2.29	1.91	1.66	1.43

であるがこれに対する近似式として土木技術資料5巻5号「F表t表5%点の近似式」によれば

$$F_{0.05} \div \left(a + \frac{b}{\phi_2} \right)^4$$

で a , b は ϕ_1 の関数として表現され

標準手続プログラム紹介

ϕ_1	a	b
1	1.40	1.02
2	1.32	1.16
3	1.27	1.28
4	1.24	1.35
5	1.22	1.40
6	1.20	1.44
7	1.19	1.48
8	1.18	1.51
9	1.17	1.54
10	1.16	1.56
30	1.10	1.79
50	1.08	1.87
100	1.05	1.97
200	1.04	2.03
∞	1.00	2.70

と近似される。

そこで分子の自由度 $F1$ および分母の自由度 $F2$ を与えられて配列 A, B の中の対応するものを取り出し中間の自由度に対しては適当な補間をし、 $F1 > 200$ に対しては ∞ に対応する係数を用いて近似値 $FT5$ を求める関数指示子を作る。

方針としては $F1$ が 10 までは array の順に、 $F1$ が 11 以上で 200 以下のときは補間計算を行ない、200 以上に対しては ∞ の場合の係数をとることとし補間計算のために INT という関数指示子をさらにもうけることとする。

```

real procedure FT5(F1,F2);
  value F1,F2; integer F1,F2;
begin integer F,I,J; real AE;
  array A,B(1:14);
  integer array C(1:14);
  real procedure INT(F,I1);
    value F,I1; integer F,F1;
    INT:=(A(F)+(A(F+1)-A(F))*(I1-C(F)))/(C(F+1)-C(F))

```

```

      +(B(F)+(B(F+1)-B(F))*(I1-C(F))/(C(F+1)-C(F)))/
      /F2)↑4;
I:=0;
for J:=1 stop 1 until 10,30,50,100,200 do
begin
  C(I+1):=J;
  I:=I+1;
end;
I:=0;
for AE:=1.40,1.32,1.27,1.24,1.22,1.20,1.19,1.18,1.17,1.16,
      1.10,1.08,1.05,1.04,1.02,1.16,1.28,1.35,1.40,1.44,
      1.48,1.51,1.54,1.56,1.79,1.87,1.97,2.03 do
begin
  if I<14 then A(I+1):=AE
    else B(I-13):=AE;
  I:=I+1;
end;
F:=if F1≤10 then F1 else if F1≤30 then 10 else if
    F1≤30 then 11 else if F1≤50 then 12 else if F1
    ≤100 then 13 else 14;
if F1≤10 then FT5:=(A(F1)+B(F1)/F2)↑4 else if F1<200
    then FT5:=INT(F,F1)
    else FT5:=(1+2.07/F2)↑4;
end;

```

特にこのプログラムでは補間のために自由度そのものを integer array C に格納したこと、A, B に対し定数を入れるのにいちいち代入命令を書かずに for 命令によつたことが特長である。前の表に対応した計算結果は

ϕ	1	3	5	10	20	60
5	6.62	5.42	5.06	4.69	4.61	4.34
10	5.09	3.82	3.42	3.00	2.83	2.43
20	4.43	3.17	2.77	2.35	2.17	1.76
120	3.94	2.69	2.30	1.89	1.71	1.31

で実用上差支えない程度に近似しているものと考えられる。

Dean Marston 公式による土圧の算定 (project 形)

技術管理室 富田 努

§ 1 概 説

暗きよなどの地下埋設物に作用する垂直土圧の算定式に Dean Marston 公式がよく用いられるが、公式には指数を含むので、計算はかなりむづかしく今までは図表などからその略値を求めるに過ぎなかつたので、この式の解を求める procedure を作り、容易にその値を求め得るようにした。

§ 2 公 式 (project 型)

$$q = C_c * W * B_c$$

$$H < H_e \text{ のとき } C_c = \frac{e^{\pm 2K\mu \frac{H}{B_c}} - 1}{\pm 2K\mu}$$

$$H \geq H_e \text{ のとき } C_c = \frac{e^{\pm 2K\mu \frac{H_e}{B_c}} - 1}{\pm 2K\mu} + \left(\frac{H}{B_c} - \frac{H_e}{B_c} \right) e^{\pm 2K\mu \frac{H_e}{B_c}}$$

たゞし H_e は次式で求める。

$$e^{\pm 2K\mu \frac{H_e}{B_c}} \pm 2K\mu \frac{H_e}{B_c} = \pm 2K\mu * r * d * p + 1$$

q = 暗きよなどに働らく土圧 (t/m^2)

W = 土の単位体積重量 (t/m^3)

B_c = 暗きよなどの外巾 (m)

K = ランキンの係数 $\left(= \frac{\mu^2 + 1 - \mu}{\mu^2 + 1 + \mu} \right)$

μ = 土の内部まさつ角

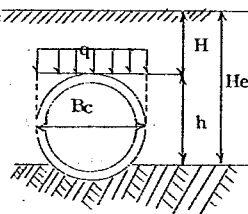
H = 土かぶり厚 (m)

H_c = 仮定の沈下面より暗きよなどの上端までの深さ (m)

p = 突出比, 盛土基面上から暗きよ上端までの高さ (h) と暗きよなどの外径 (B_c) との比 (h/B_c)

rsd = 沈下比

土 = $\left\{ \begin{array}{l} + ; \text{コンクリート暗きよなどのよ} \\ \quad \text{うに変形しにくい暗きよの場合} \\ - ; \text{コルゲートパイプなどのよう} \\ \quad \text{に変形しやすい暗きよの場合} \end{array} \right.$



§ 3 procedure について

procedure value につぎの値をいれると PH に土圧 (t/m^2) を印字する。

PM : コンクリート暗きよなどのように変形しにくい暗きよのときは 1 をいれ, コルゲートパイプなどのように変形しやすい暗きよのときは 1 以外の数字。

BC : 暗きよの外径 (管の外径) (m);

H : 土かぶり厚 (m);

F : 突出比

W : 土の単位重量 (t/m^3);

A : 土の内部まっさつ角 ($^\circ$);

B : " ($'$);

procedure $MARPR(PM, BC, H, F, R, W, A, B, PH)$;

comment

Purpose: Dean Marston Kosiki niyori project gata no

ankyo ni kuwawaru doatu o motomu,

PM : concrete ankyo no yoni henkei sinikui toki ni
wa 1, corugate nado wa 2,

BC : kan no gaikei (m),

H : dokaburiatu (m),

標準手続プログラム紹介

```

      F:tooshutuhi,
      R:tinkahi,
      W:tuti no tan i jyuryo(kg/m3),
      S:kaso no tinkamen yori kankyo no jotan madeno
        hukasa(m),
      A,B:tuti no naibumasatukaku(do,hun),
      PH:doatu(kg/m2);
value PM,BC,H,F,R,W,A,B;
real BC,H,F,R,W,A,B,PH;
integer PM;
begin real Q,K,SS,RAD,S;integer I;
  switch SW:=SW1,SW2,SW3;
    B:=B/60.0; RAD:=.14159*(A+B)/180.0;
    RAD:=SIN(RAD)/COS(RAD); K:=(SQRT(RAD2+1.0)-RAD)/
      (SQRT(RAD2+1.0)+RAD);
    Q:=0.0; I:=1;
  SW1: Q:=Q+0.1; go to START;
  SW2: Q:=Q+0.01; go to START;
  SW3: Q:=Q+0.001; go to START;
  START: if PM=1 then SS:=EXP(2.0*K*RAD*Q)-2.0*K*RAD*Q-2.0
    *K*RAD*R*F-1.0 else
    SS:=EXP(-2.0*K*RAD*Q)+2.0*K*RAD*Q+2.0*K*RAD*R*
    F-1.0;
  if SS<0.0 then go to SW(I) else
  if SS>0.0 then
    begin if I=1 then begin I:=2;Q:=Q-0.1; go to
      SW(I); end else
      if I=2 then begin I:=3;Q:=Q-0.01; go to
        SW(I); end;
    end;
end;

```

```

S:=BC*Q;
if H>S then
begin if PM=1 then
  PH:=(EXP(2.0*K*RAD*Q)-1.0)/(2.0*K*RAD)+(H/BC-Q)*EXP
    (2.0*K*RAD*Q))*W*BC  else
  PH:=(EXP(-2.0*K*RAD*Q)-1.0)/(-2.0*K*RAD)+(H/BC-Q)*
    EXP(-2.0*K*RAD*Q))*W*BC; end else
begin if PM=1 then
  PH:=(EXP(2.0*K*RAD*H/BC)-1.0)/(2.0*K*RAD))*W*BC
  else
  PH:=(EXP(-2.0*K*RAD*H/BC)-1.0)/(-2.0*K*RAD))*W*BC;
end;
PRINT(5.1,PH);CRLF(1);
end;

```

パワースペクトル計算のためのプログラム

橋本 宏*

自己相関関数とパワースペクトルを求める標準手続を示す。

1. Δt 間隔で読み取ったデータを $x(n) = x(n\Delta t)$ ($n=1, 2, \dots, M$) で表すと相関関数は

$$C_{xx}(l) = \frac{1}{M} \sum_{n=1}^{M-1} \bar{x}(l+n) \bar{x}(n)$$

$$l = 0, 1, \dots, h$$

ここで, M : データの総数, $\bar{x}(n) = x(n) - \bar{x}$,

$$\bar{x} = \frac{1}{M} \sum_{n=1}^M x(n), \quad h: \text{時間差}$$

自己相関関数のフーリエ変換によつてパワースペクトルを求める。

$$\begin{aligned} \bar{p}\left(\frac{r}{h} \cdot \frac{1}{2\Delta t}\right) &= \Delta t \left\{ C_{xx}(0) + 2 \sum_{l=1}^{h-1} C_{xx}(l) \cos\left(2\pi \frac{r}{2h} l\right) \right. \\ &\quad \left. + (-1)^r C_{xx}(h) \right\} \end{aligned}$$

$$r = 0, 1, 2, \dots, h$$

これにウィンドをかけて平均化を行なう。

$$\bar{p}\left(\frac{r}{h} \cdot \frac{1}{2\Delta t}\right) = \sum_{n=-k}^k a_n \bar{p}\left(\frac{r-n}{h} \cdot \frac{1}{2\Delta t}\right),$$

$$r = 0, 1, 2, \dots, h$$

ここで,

$$\begin{aligned} \bar{p}\left(\frac{-r}{h} \cdot \frac{1}{2\Delta t}\right) &= \bar{p}\left(\frac{r}{h} \cdot \frac{1}{2\Delta t}\right), \\ \bar{p}\left(\frac{h+r}{h} \cdot \frac{1}{2\Delta t}\right) &= \bar{p}\left(\frac{h-r}{h} \cdot \frac{1}{2\Delta t}\right) \end{aligned}$$

また, ウィンドとしては次の3種類を用いる。

* 海岸研究室

	a_0	$a_1=a-1$	$a_2=a-2$	$a_3=a-3$
W_1	(0.5132	0.2434	*	*)
W_2	(0.6398	0.2401	-0.0600	*)
W_3	(0.7029	0.2228	-0.0891	0.0149)

2. 計算条件, 自己相関関数, それぞれのウィンドウに対するパワースペクトルを印刷する。

3. パラメータについて

M : データ総数, H : 時間差, DT : 時間間隔,
 X : データ用 real array, H は 0~100 までの数

(参考文献)

H. AKAIKE; Statistical measurement of frequency response function, studies on the Statistical estimation of frequency response functions. (reprinted from Annals of the Institute of Statistical Mathematics, Supplement II, 1964)

4. プログラム

```

procedure SPECTRUM(M, H, DT, X);
  value M, H, DT;
  array X;
  real DT;
  integer M, H;
  begin real S;
    integer N, L;
    array CXX(0:100), PXX(-3:103), PXXW1(0:
      100), PXXW2(0:100), PXXW3(0:100),
      W1(-3:3), W2(-3:3), W3(-3:3);
    CRLF(5);
    LL1: PRINT('POWER##SPECTRUM');
```

標準手続プログラム紹介

```

      CRLF(2);
      PRINT('KEISAN#JOKEN'); SPACE(4); PRINT
      ('M'); PRINT(4,M); SPACE(2); PRINT('H');
      PRINT(3,H); SPACE(2); PRINT('DT'); PRINT
      (3,DT);
      CRLF(2);
      S := 0.0;
      LL2: for N := 1 step 1 until M do S := S + X[N];
            for N := 1 step 1 until M do X[N] := X[N] -
            for L := 0 step 1 until H do      S / FLOAT(M);
            begin S := 0.0;
              for N := 1 step 1 until M - L do
                S := S + X[L + N] * X[N];
                CXX[L] := S / FLOAT(M);
            end;
      LL3: PRINT('AUTOCOVARIANCE##FUNCTION');
      CRLF(2);
      SPACE(3); PRINT('L'); SPACE(3); PRINT('CXX
      (L)'); SPACE(3); PRINT('-1.0'); SPACE(6);
      PRINT('-0.8'); SPACE(6); PRINT('-0.6');
      SPACE(6); PRINT('-0.4'); SPACE(6); PRINT
      ('-0.2'); SPACE(8); PRINT('0'); SPACE(8);
      PRINT('0.2'); SPACE(7); PRINT('0.4'),
      SPACE(7); PRINT('0.6'); SPACE(7); PRINT
      ('0.8'); SPACE(7); PRINT('1.0');
      CRLF(1);
      L := 0.0;
      LL4: PRINT(2,L); PRINT(3,CXX[L]);
      SPACE(FIX(CXX[L] * 50.0 / CXX[0] + 0.5) + 52);
      PRINT('*');

```

```

CRLF(1);
L:=L+1;
if L≤H then go to LL4;
LL5; for N:=0.0 step 1 until H do
begin S:=0.0;
  for L:=1 step 1 until H-1 do
    S:=CXX(L)*COS(3.142*FLOAT(N)*FLOAT
      (L)/FLOAT(H))+S;
    PXX(N):=DT*(CXX(0)+2.0*S+(-1.0)↑N*
      CXX(H));
  end;
LL6; for N:=1 step 1 until 3 do
begin
  PXX(-N):=PXX(N);
  PXX(H+N):=PXX(H-N);
end;
W1(0):=0.5132; W1(1):=W1(-1):=0.2434;
W1(2):=W1(-2):=W1(3):=W1(-3):=0.0;
W2(0):=0.6398; W2(1):=W2(-1):=0.2401;
W2(2):=W2(-2):=-0.0600; W2(3):=W2(-3):=0.0;
W3(0):=0.7029; W3(1):=W3(-1):=0.2228;
W3(2):=W3(-2):=-0.0891; W3(3):=W3(-3):=0.0149;
LL7; for N:=0 step 1 until H do
begin S:=0.0;
  for L:=3 step 1 until 3 do
    S:=S+W1(L)*PXX(N-L);
    PXXW1(N):=S; S:=0.0;
  for L:=3 step 1 until 3 do
    S:=S+W2(L)*PXX(N-L);
    PXXW2(N):=S; S:=0.0;

```

標準手続プログラム紹介

```

        for L:=-3 step 1 until 3 do
            S:=S+W3 {L}*PXX {N-L};
            PXXW3 {N}:=S;
        end;
LL8; PRINT('POWER## SPECTRUM'); CRLF(2);
    PRINT('###R/(H*2 DT)'); SPACE(4);
    PRINT('PXX(N)'); SPACE(4); PRINT('PXXW1(N)');
    SPACE(3); PRINT('PXXW2(N)'); SPACE(3);
    PRINT('PXXW3(N)');
    CRLF(2);
LL9; for L:=0 step 1 until H do
    begin
        SPACE(2); PRINT(3,FLOAT(L)/FLOAT(H)/
        (2.0*DT)); PRINT(3,PXX {L}); PRINT(3,PXXW1
        {L}); PRINT(3,PXXW2{L}); PRINT(3,PXXW3{L});
        CRLF(1);
    end;
end;
```

電算室よりのお知らせ

§ 3 電算室よりのお知らせ

(1) プログラマー登録番号追加一覧表 (その2)

所 属	記号	先号掲載人員	研修人員	登録者人員
土木研究所本所	A	30	1	31
〃 千葉支所	B	76	3	79
〃 赤羽分室	C	52	3	55
〃 篠崎分室	D	9		9
〃 鹿島水理試験所	E	1	1	2
〃 新潟試験所	F	2		2
小 計		170		178
本省関係	G	68		68
地建関係		89		118
東北地建	K	8		8
北陸地建	L	13	5	18
関東地建	M	32	1	33
中部地建	N	8	5	13
近畿地建	P	16		16
中国地建	Q	7	18	25
四国地建	R	3		3
九州地建	S	2		2
都府県関係	T	28		28
公団・公社	U	11		11
その他	V	13		13
小 計		52		52
合 計		379		416
研修日 S.40.10.26～10.27 (本所講堂)				
中国地建出張研修 S.40.10.18～10.20 (於中国地建)				

電算室よりのお知らせ

プログラマー追加名簿 (その2)

登録番号	氏名	所	属
A 031	長谷川 謹也	技術管理室	土木研究所
B 076	三 牧 翹	構造研究室	"
B 077	永 井 隆	"	"
B 078	斉 藤 俊三	"	"
C 054	西 田 哲夫	下水道研究室	"
C 055	川 島 紘	砂防研究室	"
C 056	松 永 正昭	"	"
E 002	関 友 春	鹿島水理試験所	"
L 014	須 山 洋	河川計画課	北陸地建
L 015	柳 沢 正	企画室	"
L 016	須 田 公男	外山工事々務所	"
L 017	村 上 修	新潟国道工事々務所	"
L 018	木 村 良一	道路管理課	"
M 033	井 組 孝	東京国道工事々務所	関東地建
N 009	西 岡 正	矢 作 タ ム	中部地建
N 010	富 谷 雄	名四国道工事々務所	"
N 011	夫 馬 毅	企画室	"
N 012	佐 野 喜代志	"	"
N 013	川 上 晃 示	名四国道工事々務所	"
Q 008	赤 木 正文	鳥取工事々務所	中国地建
Q 009	山 岸 裕之	河川計画課	"
Q 010	田 中 康男	道路計画課	"
Q 011	荒 木 寛	企画室	"

電算室よりのお知らせ

登録番号	氏名	所	属
Q 012	沢山民季	企画課	中国地建
Q 013	木下良治	"	"
Q 014	青戸繁	"	"
Q 015	河村信武	"	"
Q 016	広中徹	"	"
Q 017	宮倉文雄	"	"
Q 018	森安敏博	道路計画課	"
Q 019	進藤泰啓	"	"
Q 020	石田哲之	道路工事課	"
Q 021	古本昇	"	"
Q 022	関野務	"	"
Q 023	田村末次	河川管理課	"
Q 024	円井健史	河川計画課	"
Q 025	国原五十夫	河川管理課	"

電算室よりのお知らせ

〔2〕 標準手続追加一覧表

ここに掲載の標準手続は10月30日沖電気より届き、内容については整理が出来ていないので第4号に掲載するとして標準手続名だけお知らせします。

登録番号	手 続 名	
EX 1061	任意の実数の階乗の逆数 $1/n!$	INVFACTORIAL
EX 1062	任意の実数の階乗 $n!$	FACTORIAL
EX 1063	ベッセル関数(円柱関数) $J_n(x)$	BESSEL
EX 1064	データプロット	PLÖT
EX 1065	ガンマ関数 $\gamma(x)$	GAMMA
EX 1066	準有理式近似	RAT APPRO
EX 1067	対称行列の固有値及び固有ベクトル(Power)	UNPÖW
EX 1068	" (Jacobi)	JACÖBI
EX 1069	行列の特性方程式係数(Danil'ski)	DANILV
EX 1070	" (Frame)	FRAME
EX 1071	三次方程式	PÖLY3
EX 1072	常微分方程式	RUNGE KUTTA
EX 1073	"	MILNE
EX 1074	連立常微分方程式	SDRKG
EX 1075	連立微分方程式	SDEULER
EX 1076	" (Merson)	SDMERSÖN
EX 1077	常微分方程式 (Merson)	MERSÖN
EX 1078	複素数演算	CÖP
EX 1079	ベクトル外積	VEXP
EX 1080	複素変数の指定関数	EXPC
EX 1081	素 数	PRIME