

44—8

電 子 計 算

技 術 管 理

土 木 研 究 所 資 料

第 479 号

電 算 速 報

第 10 号

昭 和 44 年 8 月

建 設 省 土 木 研 究 所

電 算 速 報 1 0 号 目 次

§ 1. アルゴルプログラマのための PL/I 入門	1
§ 2. ALGOL, PL/I, FORTRAN, COBOL の比較	19
§ 3. 自動図化とプログラム例	29
§ 4. 電算室よりのお知らせ	37
〔1〕 電子計算機稼働表	37
〔2〕 プログラム名簿	40
〔3〕 TOPS-14 T-ALGOL ERROR TABLE	41

§ 1. アルゴリズムのための PL/I 入門

中 村 慶 一

1963年IBMのユーザ団体SHAREおよびIBM社員よりなる委員会の手によってALGOLのブロック構造、混合演算、繰返し文や条件文の多様性、手続宣言などを全面的に取り入れ、それにCOBOLのデータ構造やデータ記述を取り入れ、さらに配列の配列といった概念を付加した言語の開発を始め、1965年PL/Iとして発表した。その記述形式もALGOLにならいカードの欄などには全く拘束されることなく、文の終りに;を付しIF THEN ELSEをはじめWHILEや=><などの使い方もALGOLの場合と同様なので次の諸点さえ注意すればALOLを知っているものもすぐにもALGOLと同じ内容のPL/Iプログラムを書き下すことが出来る。(ただし残念なことに条件式は使用出来ない)

また データ、構造、データ記述、配列の配列等の知識を追加することによって、より多様なPL/Iプログラムを書くことが出来るので以下これらについて触れてみたい。

A ALGOLを書き直す場合の注意

1. ブロック

ALGOLではbegin end という文の括弧の中で宣言が行なわれることによってブロックを構成しproceduer 宣言では暗黙のうちにブロックが作られることになっていたがPL/IではBEGINステートメントとENDステートメント、あるいはPROCEDUREステートメントによってブロックが構成される。その機能は全く同じであるがALGOLはプログラム全体が1つのブロックあるいは複合文であったのに対しPL/Iではプログラム全体が1つのPROCEDUREブロックを構成しこの呼び出しによって実行する代りにプログラムは

名 前 : PROCEDURE OPTIONS (MAIN);

.....

.....

END;

という形を取りこの場合に限 りCALL文なしに本体の実行を行なう。なおALGOLの複合文に当るものにDOステートメントによって作られるDO

グループがあり

D $\bar{O}$:

.....

.....

END :

となる。(くり返し文としてのD $\bar{O}$ は後にのべる)

PROCEDUREはPROCと略しても良い。

2 I/ $\bar{O}$ 文

READ(A, B, C): は GET LIST(A, B, C);

PRINTI(5, E): は PUT EDIT(E)(F(6), x(2));

PRINTR(5, E): は PUT EDIT(E)(E(11, 5), x(2));

PRINTE(32, E): は PUT EDIT(E)(F(7, 2), x(2));

SPACE(3): は PUT EDIT(x(3));

CRLF(2): は PUT SKIP(2);

PRINTS('ABC') は PUT LIST('ABC');

NEWPAGE: は PUT PAGE;

とすればよい。もちろんリストフォーマットはまとめてPUT EDIT('MEAN=')(SKIP, A, F(7, 2)); などとしてよい。もちろん標準フォーマットでよければPUT LIST(A, B, C); でよいし、読み込みにフォーマットをつけたければ、GET EDIT(リスト)(フォーマット); とする。

3 条件文

ALGOLの

if A=B then

begin if C $\neq$ D then S:=S+1 else S:=S-1 end

else R:=P*Q;

はPL/Iでは

IF A=B THEN

IF >C]T=D THEN S=S+1; ELSE S=S-1;

ELSE R=P*Q;

となる。つまりALGOLではelseの前の;は絶対に入れてはならないことになっていたが、PL/Iでは必ず入れなければならない。したがって

```

if A=B then begin if C≠D then S:=S+1 end else
  R:=P*Q;

```

はALGOLと同じ空文を用いて

```

IF A=B THEN IF C=D THEN S:=S+1;ELSE;
  ELSE R=P*Q;

```

とする。なお代入記号の=は比較作用素と共用される。

また、条件式はないから条件文に直さなければならない。

4. 繰り返し文

```

for A:=3 step 47 until 50, A*2 while A<350 do
  PRINTR(5, A);

```

は

```

DÖ A=3 BY 47 TÖ 50, A*2 WHILE(A<350);
A=3 PUT EDIT(A)(E(11,5));END;

```

とする。DÖ ステートメント自身が;で区切られることはALGOLではdo;となって空文を繰り返すことになるので厳禁されるがPL/Iでは必ず;を入れなければならない。また必ずEND ステートメントを必要とする。

きざみが1のときはBY1を省略してよい。また、BYとTÖはどちらが先になってもよい。また、DÖ WHILE(論理式);という形も許される。さらに1でのべたようにDÖ;も使用できる。

5. 宣言

ALGOLと同じようにすべての名前は宣言を必要とする(指定によりI~Nで始まると整数、以外は実数とみなす)

ALGOLで

```

begin
  real A,B,C; integer I,J;
  array AR [-3:5];
  switch SW:=L1,L2,L3;
  .....
  go to SW [1];
  L1: .....
  L2: .....
end;

```

は PL/I では

```
KEISAN:PROCEDURE OPTIONS(MAIN);DECLARE
(A,B,C)FLOAT (I,J)FIXED,((AR,S)(-3:5))
FLOAT,SW(3) LABEL INITIAL(L1,L2,L3);
```

.....

```
GO TO SW(1);
```

```
L1:.....
```

```
L2:.....
```

.....

```
END;
```

となる。FLOAT, FIXEDの代りにDECIMAL, BINARYでもよい。

名前の後に書かれるFLOAT, FIXED, LABEL等を属性と呼ぶが
PL/Iではこのほかにalphaに当るCHARACTER(n)などもある。ま
た()でつつんでまとめ書きをすることもできるし、配列で下限1のとき
は1:を略してもよい。宣言は同じレベルならどこにおいてもよいが、修
正時の便を考えると先頭に集めておいた方がよい。

6 初期値設定

KDK ALGOLで

```
real A,B initial 0;
```

としたものはPL/Iでは

```
DCL(A,B)FLOAT INITIAL(0);
```

である。ただし、DCLはDECLARAEの省略形でこれを用いてもよい。

KDK ALGOLでベクトル式を用い

```
array A[1:3] initial vec 3 8 4 5 6 5 tor;
```

はPL/Iでは

```
DCL A(3)FLOAT INITIAL(38,45,65);
```

である。

7 手続き

手続き(関数を含む)宣言はPROCEDUREステートメントとEND
ステートメントでかこまれたPROCEDUREブロックによる。

```
procedure SUM(A,B,C);
```

```
real A,B,C;
```

```
C:=A+B;
```

は

```
SUM: PROCEDURE(A, B, C);  
  DCL(A, B, C) FLOAT;  
  C=A+B;  
  END;
```

であり

```
real procedure SUM(A, B);  
  real A, B;  
  SUM:=A+B;
```

は

```
SUM: PROCEDURE(A, B) FLOAT;  
  DCL A, B FLOAT;  
  RETURN(A+B);  
  END;
```

とする。

全く形式的な置換えであるがただ ALGOL の規制に当るものではなく、この DCL は仮パラメタを含みすべてこの PROCEDURE ブロック内限りで使用される名前の宣言である。

手続きの実行は ALGOL の SUM(P, Q, R); に当るものが PL/I の CALL SUM(P, Q, R); である。関数は ALGOL と全く同じように式の中で用いる。

8 多重代入

```
ALGOL  A:=B:=C:=0;  
PL/I  A, B, C=0;
```

9 配列の配列

KDK ALGOL では

```
A[*,*]:=vec vec 32,45,68 tor,vec 29,31,15 tor tor;  
A[I,*]:=vec 14,32,11,18,15 tor; A:=vec 0 tor;
```

というベクトル式を認めたが PL/I ではさらに多様化して DCL(X, Y)
(2, 3), (U, V) (3, 2) FLOAT; に対して

X=2*Y; によって

```
X(1, 1)=2*Y(1, 1); X(1, 2)=2*Y(1, 2);
```

```

X(1,3)=2*Y(1,3);
X(2,1)=2*Y(2,1); X(2,2)=2*Y(2,2);
X(2,3)=2*Y(2,3);

```

のすべてを実行させ得るし

```

U(*,1)=2*Y(1,*); によって
U(1,1)=2*Y(1,1); U(2,1)=2*Y(1,2);
U(3,1)=2*Y(1,3);

```

のすべてを実行させることも可能である。したがって同じ階数の配列 A, B, C で $C = A + B$; なども可能であるが, $C = A * B$; や $C = A / B$; は対応要素の積または商でマトリクス演算にはならない。

10. 読込書出の反復指定

FÖRTTRAN にならって

```

PUT EDIT((A(I,J) DÖ I=1 TÖ 3) DÖ T=1 TÖ 4)
      (SKIP, 2P'ZZ9 ');

```

という反復指定が使用できる。ただし P はフォーマット形式を示し Z はゼロサブレスする桁 9 はゼロサブレスしない桁を示す。

また配列の取り扱いが ALGÖL にならって行なりに行なう。さらに前の * を用いて

```

PUT EDIT(A(*,K) DÖ K=1 TÖ 4) (SKIP, 2P'ZZ9 ');

```

という使い方も可能である。

11. 組み込み関数

ABS, EXP, LOG, ATAN, SIN, COS, SQRT などのほかに MAX(A,B), MIN(A,B), SUBSTR(S,M,N); INDEX(S1, S2) などがある。S は文字ストリングでストリング中の M 番から N 番までもし N がなければ M 番目から後の全部のストリング, S1, S2 も文字ストリングで S2 が S1 中の何番目から始まっているかという番号を示す。たとえば

```

DCL X CHARACTER(6);
X='STRÖNG';

```

で INDEX(X, 'S') は 1 を INDEX(X, 'O') は 4 をあらわす。また SUBSTR(X, 1, 3) は 'STR' を SUBSTR(X, 3, 5) は 'RÖN' をあらわすことになる。

さらに前にのべた*を用い

$$X = \begin{bmatrix} 3, & -1, & 3 \\ 0, & 2, & 3 \end{bmatrix} \quad Y = \begin{bmatrix} -1, & 0, & 2 \\ 1, & 4, & 5 \end{bmatrix}$$

に対して

```
DECLARE  X(2,3) FLOAT INITIAL(3,-1,4,0,2,3),
Y(2,3)  FLOAT INITIAL(-1,0,2,1,4,5),
(Z(2,3),R1(2),R2(3)) FLOAT;
```

としてあれば

$$Z = \text{MAX}(X, Y); \text{ によって } Z = \begin{bmatrix} 3, & 0, & 4 \\ 1, & 4, & 5 \end{bmatrix}$$

$$Z = \text{MIN}(X, Y); \text{ によって } Z = \begin{bmatrix} -1, & -1, & 2 \\ 0, & 2, & 3 \end{bmatrix}$$

となるほか

$$R1 = X(2,*) + Y(1,*); \text{ によって } R1 = (-1, 2, 5)$$

$$R2 = \text{MAX}(X(*,2), Y(*,3)); \text{ によって}$$

$$R2 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$$

となる。

12 ENDの省略

```
L1: DO I=1 TO M;
    DO J=1 TO N;
        BEGIN;
        DCL AE FLOAT;
        AE=0;
        DO K=1 TO L;
            AE=AE+A(I,K)*B(K,J);
        END;
        C(I,J)=AE;
    END L1;
```

のようにENDの後にレーベルを書くと、そのレーベル以降で直前に必要なENDで不足しているものはすべて補なわれる。

13 データの構造

COBOLにない宣言時にレベル番号を付してDCL(1X, 2Y, 2Z, 2R, 3S, 3P, 1A, 2B, 2C, 2D, 3E, 3Q) FLOAT;

に対し

$X = Y * A$; によって

$Y = Y * B$; $Z = Z * C$; $S = S * E$; $P = P * Q$

またこれは配列に関しても

$DCL(1A(2, 2), 2B, 2C, 2D, 3E, 3F, 1R, 3S, 3T, 3U,$
 $5V, 5W) FLOAT;$

に対し

$A = R$; は

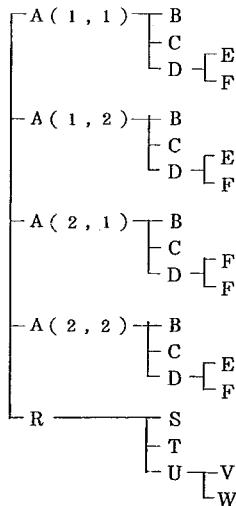
$A(1, 1).B, A(1, 2).B, A(2, 1).B, A(2, 2).B = S$;

$A(1, 1).C, A(1, 2).C, A(2, 1).C, A(2, 2).C = T$;

$A(1, 1).E, A(1, 2).E, A(2, 1).E, A(2, 2).E = V$;

$A(1, 1).F, A(1, 2).F, A(2, 1).F, A(2, 2).F = W$;

の全てを意味することになる。名前は左から右にレベル番号が増す順にピリオドでつないだ書き方で修飾することができる。上の場合



という構造をもつのでこれらを識別するため上記のような表現を用いる。

14. 表 現

今までの例でも分る通り PL/I ではデリミタに大文字を使うが ALGOL とまったく同じように用いる記号として ; , / () + - > < がある。= は ALGOL の : = と = の両様に用いる。] = > = < = は ALGOL の $\niq$ $\geq$ $\leq$ の意味であり、& | は ALGOL の $\wedge \vee$ の意味に用いられる。また * は掛算であるが ALGOL の $\uparrow$ の代りに ** が用いられる。添字を示す [] はすべて () に置き換えられる。ALGOL で用いる引用符 ' ' は PL/I では ' を共用する。
or を意味する | は || と用いることによつて連結という概念を表示する。これはストリングの連結で 'PO' || 'AB' は 'PQAB' に等しい。
なお ALGOL の 36.210-3 は 36.2E-3 と書かれるが ALGOL では禁示されていた 4. という表現も許される。

15. コメント

ALGOL の comment ; や end の後のコメントは PL/I では / * * / とすればよくブランクを挿入できるところならどこでもよい。

16. PICTURE

PUT, GET ステートメントのデータリストに指定した変数のピクチャーを示すもので P ' ' という形で表現する。

9 : 10 進法の桁位置

V : 仮想小数点の位置 (小数点は入らない)

Z : ゼロサプレスする桁

B : ブランク

, : カンマ

. : 小数点

S : 符号

- : 符号 (負の時のみ)

たとえば 1 2 3 4 に対し 9 9 9 9 9 を指定すると 0 1 2 3 4 Z Z Z B

9 9 なら 1 2 3 4 となる。

17. 代入ステートメントと擬変数

ALGOL で代入文は変数 (単純又は添字付) : = 式 : という形により特例として手続宣言の中で関数の場合に、手続き名 : = 式 ; であったが、

PL/I では

スカラ変数=スカラ式;

擬 変 数=文 字 式;

配 列 名=配 列 名;

構 造=構 造;

名札変数=名 札;

名札配列=名札または名札変数または名札配列;

などがある。もちろん多重代入の時は左辺はカンマでつないで良い。

擬変数としてはSUBSTR, COMPLEX, REAL, IMAGなどがある。

```
DCL X CHARACTER(6);
```

```
X='STRÖNG';
```

のとき

```
SUBSTR(X, 2, 4)='AYI';
```

によつてXの値は“SAYING”となる。

18 論理変数

ALGÖLの

```
Boolean K; Boolean array SA[1:20, 0:2];
```

は

```
DCL(K, SA(1:20, 0:2)) BIT(1);
```

となり論理値の代入読み込みのほか、TRUE が1 FALSEが0に対応しているので算術式の中に持ちこんでもよい。

19 手続きの仮パラメタとしての手続き名

ALGÖLの

```
procedure AL(A, B);
```

```
  procedure A; real B;
```

```
    bgin
```

```
      real C;
```

```
      .....
```

```
    end;
```

は

```
AL: PROC(A, C);
```

```
  DCL (B, C) FLOAT, A ENTRY;
```

```
| .....  
END;
```

で、Aが2つのパラメタを持ち、1つがreal次がintegerのとき

A:ENTRY(FLOAT, FIXED):とすると効率が良い。

FORTRANのSUBROUTINE処理のようにプログラム本体と別にコンパイルして呼び出すときも同様にENTRY属性を指定すればCALLできる。

20 仮パラメタとしての配列名

ALGOLの

```
procedure SUM(A, B, M, N);  
  array A, B, C: integer M, N;  
  begin  
    integer I, J, K; real AE;  
    .....  
  end;
```

はA, B, C共に2次元なら

```
SUM:PROC(A, B, C, M, N);  
  DCL((A, B, C)(*, *), AE)FLOAT, (M, N, I, J, K)  
  FIXED;  
  .....  
END;
```

B アルゴリズムよりPL/Iへの変換

P1 紙テープに個数Nを先頭にN個のデータが入っている。これを読んで最大値、最小値を求め次の形で印字せよ(有効6桁で印字)。

MAX=.....

MIN=.....

P2 任意形の面積を求めるのに矩形と三角形に区分して計算し後で合計するプログラムをつくれ。

ただし、各辺の長さのみしか測定できなかったとし、各々の面積を計算する部分はprocedureを使用すること。

長方形 $A = L_2 \times L_3$

$$\text{三角形} \quad S = \frac{1}{2} (L_1 + L_2 + L_3)$$

$$A = \sqrt{S(S-L_1)(S-L_2)(S-L_3)}$$

P3 N個のデータ (A[I]) があたえられたとき、このデータを大きさの順 (B[I]) にならべかえ、それが何番目のデータ (C[I]) かを求める手続きをつくれ。

P4 1組のデータ $X_1, X_2, \dots, X_n$ のヒストグラムを求める手続きをつくる。ただし区間を D_1, D_K とし、それを K 等分するものとする。

I アルゴリズムプログラム

1. begin

```
integer I, N;
READ(N);
begin array A[1:N];
  real MAX, MIN;
  READARRAY(A, 1, N);
  MAX:=MIN:=A[1];
  for I:=2 step 1 until N do
    begin real P;
      P:=A[I];
      if P>MAX then MAX:=P;
      if P<MIN then MIN:=P;
    end;
  PRINTS('MAX='); PRINTR(6, MAX);
  CRLF(1);
  PRINTS('MIN='); PRINTR(6, MIN);
  CRLF(1);
end;
end;
```

2. begin real SA, A, L1, L2, L3;

```
procedure AREA(D1, D2, D3, A);
  real D1, D2, D3, A;
  begin real S;
    if D1≠0 then go to LL;
    A:=D2*D3; go to END;
  LL: S:=(D1+D2+D3)/2;
    A:=SQRT(S*(S-D1)*(S-D2)*(S-D3));
  END;
end;
SA:=0;
ST:READ(L1, L2, L3);
if L1=0∧L2=0∧L3=0 then go to OWARI;
AREA(L1, L2, L3, A);
SA:=SA+A; go to ST;
OWARI:
PRINTS('SA='); PRINTR(5, SA);
CRLF(1);
end;
```

3. procedure ORDER(N, A, B, C);

```
integer N; array A, B;
integer array C;
begin integer I, J;
  for I:=1 step 1 until N do
    begin C[I]:=I;
      B[I]:=A[I];
    end;
  for I:=1 step 1 until N-1 do
    for J:=I+1 step 1 until N do
      if B[J]>B[I] then
        begin real P; integer M;
          P:=B[I]; M:=C[I];
          B[I]:=B[J]; C[I]:=C[J];
          B[J]:=P; C[J]:=M;
        end;
    end;
end;
```

4. procedure HSTGRM(N, D1, DK, K, X, A);

```
integer N, K; real D0, DK; array X;
integer array A;
begin integer I, J;
  real H;
  procedure CUNT(X, Y);
    real X, Y;
    if X≤Y then
      begin A[J]:=A[J]+1; go to END; end;
  ※ A[*]:=dec 0 for;
  H:=(DK-D1)/K;
  for I:=1 step 1 until N do
    begin
      for J:=1 step 1 until K do
        CUNT(X[I], D1+J*H);
      END;
    end;
end;
```

※ KDK-ALGOLによる

```
for I:=1 step 1 until N do A[I]:=0;
と同じ効果である。
```

II PL/I プログラム

```

P1: PROCEDURE OPTIONS(MAIN);
    DECLARE (I,N)FIXED BINARY;
    GFT LIST(N);
    BEGIN;
        DCL (A(N),MAX,MIN)FLOAT;
        GET LIST(A);
        MAX,MIN=A(1);
        DO I=2 TO N;
            BEGIN;
                DCL P FLOAT;
                P=A(I);
                IF P>MAX THEN MAX=P;
                IF P<MIN THEN MIN=P;
            END;
        END;
        PUT EDTT(MAX,MIN)
            (' MAX= ', F(11.6),
             X(2), SKIP, ' MIN= ',
             F(11.6)SKIP);
    END;
END P1;

```

```

P2: PROC OPTIONS (MAIN);
    DCL (SA,A,L1,L2,L3) FLOAT;
    AREA : PROC(D1,D2,D3,A);
        DCL (D1,D2,D3,A,S) FLOAT;
        IF D1 1=0 THEN GO TO LL;
        A=D2*D3; GO TO END1;
        LL: S=(D1+D2+D3)/2;
        A=SQRT(S*(S-D1)*(S-D2)*(S-D3));
    END1:

```

```

1
END;
SA=0;
ST: GET LIST(L1,L2,L3);
    IF L1=0 & L2=0 & L3=0 THEN
        GO TO OWARI;
    CALL AREA(L1,L2,L3,A);
    SA=SA+A; GO TO ST;
OWARI: PUT EDIT(SA)('SA=',F(10,5),SKIP);
END P2;

```

```

ORDER: PROC(N,A,B,C);
| DCL (I,N,J,C(*))FIXED BINARY,
|   ((A,B)(*))FLOAT;
| DO I=1 TO N;
|   | C(I)=I; B(I)=A(I);
|   END;
| DO I=1 TO N-1;
|   | DO J=I+1 TO N;
|   |   | IF B(J)>B(I) THEN
|   |   |   BEGIN;
|   |   |   | DCL PFLOAT,M FIXED BINARY;
|   |   |   | P=B(I); M=C(I);
|   |   |   | B(I)=B(J); C(I)=C(J);
|   |   |   | B(J)=P; C(J)=M;
|   |   END;
|   END;
END ORDER;

```

```

HSTGRM: PROC(N,D ,DK,K,X,A);
| DCL(I,J,N,K,A(*)) FIXED BINARY,
|   (DO,DK,X(*),H) FLOAT;
| CUNT : PROC(X,Y);
|   | DCL (X,Y) FLOAT;
|   | IF X<=Y THEN

```

```

DO;
  A(J)=A(J)+1;
  GO TO EN;
END;
A=0;
H=(DK-D1)/K;
DO I=1 TO N;
  DO J=1 TO K;
    CALL CUNT(X(I),D1+J*H);
  EN;
END HSTGRM;

```

C FORTRAN, COBOL, PL/I, ALGOLの対比

科学技術用語としての FORTRAN と ALGOL の優劣については PL/I がほとんど ALGOL を引きつぎそれに COBOL のデータの構造や PICTURE 機能をつけ加えたものになったことで、IBM みづからがはからずも軍配をあげた形になってしまった。また COBOL そのものもデータの構造という特長を除いては一般の会社や官庁の事務関係の人々の仕事に対しても大変面倒でかつ使いにくいものである。

よく言われたカード入力なるものの特長も IBM 編「FORTRAN ユーザーのための PL/I 手引書」の 9 頁で、このカードの欄に支配されるやり方はもはや妥当でないと述べているが、それを含めて 4 言語の特長を表示してみるし次のようになろう：(○は PL/I に引き継がれた特長 □は独自の特長)

	FORTRAN	COBOL	PL/I	ALGOL
1. 単語	30	318(予約語)	135	24
記号	11	15	24	30
其他	11		10	
入出力用	10	10	9	10
2. カード欄拘束	される	される	されぬ	○されぬ
3. ブロック構造	ない	ない	あり	○あり
4. 混合算	認めない	10のべきなし	認める	○認める
5. 配列次元	3(IV)	8	無制限	○無制限
6. 配列下限	1に限る	1に限る	自由	○自由
7. 添字式	C*V±Cに限る	結果正整数に限る	条件式外は自由	○自由
8. データの構造	ない	○ある	ある	ない
9. 配列の構造	ない	ない	□ある	ない(KDKはある) ○
10. 初期値設定	○ある(DATA文)	○ある(VALUE)	ある(INITIAL)	ない(KDK initial) ○
11. 複素数	○IVある	ない	ある	ない(KDKはある) ○
12. 倍精度	○IVある	ある(PICTURE)	ある	ない(KDKはある) ○
13. アルファモード	○ある	ある()	ある	ない(KDKはある) ○
14. 条件式	ない	ない	ない	□ある
15. 手続きの複入 □	○ある	ない	ある	ない
16. Bに配列AのI番目(C=DならRさもなければS番目)+3倍のA12(E≠FならA1さもなければA2)の和を代入せよ	IF(C, EQ, D) GO TO 10 I=S GO TO 50 10 I=R 50 IF(E, NE, F) GO TO 20 A12=A2 GO TO 60 20 A12=A1 60 B=A(I)+30 *A12	IF C IS EQUAL TO D MOVE R TO I ELSE MOVE S TO I. IF E IS NOT EQUAL TO F MOVE A1 TO A12 ELSE MOVE A2 TO A12. COMPUTE B=A(I)+3* A12	IF C=D THEN B=A(R); ELSE B=A(S); IF E=F THEN B=B+ 3*A1; ELSE B=B+3*A2;	B:=A[if C=D then R else S] +3*(if E≠ F then A1 else A2);

	FORTRAN	COBOL	PL/I	ALGOL
17 Aが3から 4.7きざみで 50に達する まで、次に その2倍ご とに350に 達するまで の値を有効 8けたで印 字せよ。	DO 10 I=30,500,47 AI=I/10 10 WRITE(6, 20)AI 50 IF(AI,GE, 350.0) GO TO 100 AI=AI*2.0 WRITE(6, 20)AI 20 FORMAT (1H0,E11.5) GO TO 50 100 (改行部分 はさらに別 途注意が必 要)	PERFORM IN VARYING A FROM 3 BY 4.7 UNTIL A GREATER50 ST. COMPUTE A=A*2 IF A NOT LESS 350 GO TO OWARI. IN.WRITE. PRINT FROM EN.GO TO ST OWARI. (ファイルレコ ードは定義され たものとする)	DO A=3 BY 4.7 TO 50, A*2 WHILE(A< 350); PUT EDIT (A)(E(11,5)); END;	for A:=3 step 4.7 until 50, A*2 while A<350 do PRINTR (5,A);

§ 2. ALGOL, PL/I, FORTRAN, COBOLの比較

中 村 慶 一

はじめに

よくプログラム用言語の比較が問題にされる。科学技術用はALGOL, FORTRAN 事務用はCOBOLそれらを総合したのがPL/Iなどよく言われる。ひどい場合には科学用ALGOL技術用FORTRAN などと言われることもある。科学用, 技術用, 事務用の定義がよく分らないが, 大学の人が好んで使うから科学用, 工場の人がよく使うから技術用, 事務屋が使うから事務用というのでは話がおかしくなる。事務屋さんがよく統計解折用に使うがこれは純然たる科学技術用であろうし, FORTRANは約束ごとが多いから自由を好む大学人にきらわれるというわけのものでもなかろう。

全く同じ問題をそれぞれプログラムしてその易を比較することができれば, ある程度ははっきりするので, 適当な問題をさがしていたがたまたま, 安田寿明著プログラム入門が給与計算のプログラムをはじめFORTRAN 後でCOBOLで書いているのを見て, これを後の修正の便を考えて税, 扶養などの手続きにまとめたものを4つの言語で書いてみた。

実用プログラムとしては毎月その人の本俸, 家族数なども読み込むのは, ばかして居るので台帳に何級, 何号, 家族構成などを記入しておいて, 毎月読み込むのは残業時間だけにすべきであろうが, プログラムを長くしないために, 毎月読みこむ形をとった。

なおプログラムの難易を比較してみても一度作れば良い問題については問題にならないという声もあるが, 情勢の変化に応じてプログラム修正が可能である必要がないならば全部外注して差し支えない筈で我々は来るべき情報化社会において人間機械系のパートナーとして計算機を評価するという立場に立つべきであり, そのためにはこの系における対話用語が人間に近い機械に近いかは大いに問題になる筈であると考えられる。

問題

部課コード(6けた) LBUKA

社員番号(5けた) NO

本 俸(6けた) HONPO

職務手当(5けた) SHOKMU

妻帯コード(1けた) SAI (あり 1, なし 9)

家族コード(2けた) PKAZU (妻を除く扶養家族数)

残業時間(3けた) ZTIME

の順にデータを読みこみ

- 1 残業手当 ZANGYO 1時間当たり本俸の0.5%
- 2 扶養手当 FUYO 妻 2,000円 家族一人当たり 1,000円
- 3 所得 SHOTOKU 本俸, 職手当, 家族手当, 残業手当の和
- 4 税 合 ZEIKIN
所得 12万以上 10万以上 8万以上 5万以上 3万以上 3万未満
税率% 20 18 15 10 5 0
- 5 手 取 TEDORI 所得一税金

残業時間は時間単位, 残業手当, 税金は円以下切りすてとし, 部課別にそれぞれの合計も印刷する。報告書の形式は次の通り

BUKA NO HONPO SHOKUMU FUYO ZANGYO SHOTOKU
ZEIKIN TEDORI

```
XXXXXX XXXXX XXXXX XXXXXX XXXX XXXX XXXXXX XXXXXXXX
XXXXXXX XXXXXXXX
TOTAL XXX XXXXXXX XXXXXX XXXX XXXXXX XXXXXXX XXXXXXX
      (人数)
XXXXXXX
```

方針としては

- 1 合計欄クリア KSWに初期値1代入
- 2 データ読みこみ
- 3 終りのデータなら6へ KSW 1なら2として部課コードをMに変更
- 4 部課コードが変わったら, トータルを印字し, クリアして2へ
- 5 手当計算し個人別印字と部課別合計
- 6 おわり

ALGOL

- 1 法律改正時の変更を見越して次のものは手続きにまとめ先頭におく
 - a ZEI 所得による税金計算
1案 条件式を用いる。 $ZEI := (if S \geq 120,000 \text{ then } 0.12 \text{ else } if \dots) * S - 5 ;$

2案 ベクトル式を用いる。(KDK-ALGOLの場合)

```
A := vec 120000, 100000, 80000, 50000, 30000 tor;  
B := vec 02, 018, 015, 01, 005 tor;  
ZEI := 0;  
for I := 1 step 1 until 7 do  
begin ZEI := S * B[I]; go to END; end;  
END;
```

3案 初期値設定を用いる(KDK-ALGOLの場合)

```
array A[1:5] initial vec 120000, 100000, 80000,  
50000, 30000, 02, 018, 015, 01, 005 tor;
```

として後をつづける。

いまは条件式を採用する。

- b FU 妻などの家族数による扶養手当
- c ZAN 残業時間と本俸による残業手当
- 2 表頭、印刷と合計欄のクリアをCLEARとする。
- 3 部課コードの変更によるtrueとなるブーリャン関数Bをつくる。
- 4 合計と各行の印刷は最初の2個を除き同じ形式なので、上のBを用いて配列E(各行)T(合計)の使い分けをする手続きTOTALを作る。

PL/I

- 1 手続きと関数はALGOL同様区別がなくただ関数の時は自分の名前に代入するかわりにRETURNの後の括弧内に式を書く。
- 2 条件式がないため

```
ZEI: PROCEDURE(S);  
DECLARE (A, B)(5) INITIAL(120000, 100000,  
80000, 50000, 30000, 02, 018, 015, 01, 005);  
DO I = 1 TO 5;  
IF S = A(I) THEN RETURN(S * B(I));  
END;  
RETURN(0);  
END ZEI;
```

という形をとる。

またTOTALも合計専用とせざるを得ない。

- 3 Boolean procedure に当るものがないのでそのたびに条件により分岐させざるを得ない。
- 4 ALGOL同様欄にはしばられない。
- 5 ALGOL同様文の区切りに ; を入れる。
- 6 ALGOL同様 IF THEN ELSE を (文に対してのみ) 用いるが ELSE の前には ; が必要。
- 7 ALGOL同様ブロック構造が存在し、すべての名前は宣言が必要、BEGIN ; は DO ; と同様 END ; と共に文をくくるのに用いるが、宣言は BEGIN ; の後に限る。
- 8 END ; が重なるとき END LP ; などとすると LP というレーベルまでの対応 END ; は入るものとみなされる。
なおレーベルは ALGOL 同様名前で良い。
- 9 混合演算、再帰の呼出、添字の下限は負も可、添字数無制限、添字部の書き方が自由、複合代入文 DO に WHILE の使用や正整数以外も自由、などすべて ALGOL と同様である。
- 10 KDK-ALGOL 同様初期値設定、文字型変数、A(I,*) などの使用等が可能。
- 11 $T = T + E ;$
などという配列全体を名前だけで構造体として扱い得る。
(PL / I の特長)

FORTRAN

- 1 関数 ZEI と FU, ZAN を同様つくるが、後 2 者は文関数とし実行文の前に移す。
 - a ZEI
条件式とベクトル式がないので A と B にデータ文により定数を代入して DO を用いる。
 - b FU
文のはじめに $FU = S * 20000 + P * 10000$ と混合算をさせて .0 をつける。
- 2 CLEAR
 - a 配列 T にひとつずつ代入する位ならひとつずつ THONPO などとしてこれに和をとった方が早い、ブロック構造がないので長々とパラメタ

部に書くのがいやならCOMMONとせざるを得ない。

b THONPOなどに初期値0を入れるのにALGOLでは

```
THONPO:=TSHOKM:=.....:=0;
```

という複代入文があるがFORTRANでは1つ1つ書かざるを得ない。

KDK-ALGOLでは

```
T:=vec 0 tor ;
```

または

```
array T[1:7] initiavec 0 tor ;
```

が許される。

3 TOTAL

各行と合計欄の区別によって印刷様式を変更することは条件式がないので二度書く手数と変らなくなる。したがって合計欄専用のものとする。

4 先頭から6文字分は特別な意味をもつほか、データがフォーマットに従ったものになっている必要がある。

5 論理式は算術式同様混合算を許さない。

COBOL

1 関数がないがALGOLのprocedureに似たPERFORMがあるのでこれにまとめて最後につけ変更に応じ得るようにする。

ただし、パラメタ部が設けられないので条件によってGO TOを行なうてやるより仕方がない。SUBROUTINEとちがって共通の名前はそのまま使用できるのでCOMMONの必要はない。

2 CLEAR

```
PERFORM TOTAL-CLEAR VARYING I FROM
```

```
I BY I UNTIL I GREATER 8
```

```
TOTAL-CLEAR.
```

```
MOVE ZERO TO TOTAL-WORK(I)
```

などと単に配列TOTAL-WORKの8要素に0を入れるだけでも大変でし

かもこのような1時作業部分もWORKING-STORAGE SECTION

```
で77 TOTAL-WORK PICTURE 9(10) OCCURS 8
```

```
TIMES. などとその書式指定まで必要とされる。(今は10けた)
```

3 条件式がないので

```
IF WORK-AREA(6) IS NOT LESS THAN 30000
```

```

MOVE a05 TO WORK-3.
IF WORK-AREA(6) IS NOT LESS THAN 50000
MOVE a10 TO WORK-3.

```

.....

などとひとつひとつ書かなければならない。

- 4 文は 11 欄までそれ以外は 7 欄まで特別な意味を持つほか = + - * / >
< は前後は 1 文字以上あけなければならぬとか、小数点以外の ・ や ;)
の前には空白をおいてはならず後には空白をおかなければならないとか ()
の前には 1 字以上の空白をおき、後には空白をおいてはならないといった
機械的な制約が多い。
- 5 IS NO DATA 等約 320 個の保留語があるため、いままでの社員番号
に使っていた NO が使えなくなって NA とせざるを得なかった。
- 6 データの記述で読み込み用、計算、印字用の FILE, RECORD が別々に
必要。
- 7 カナを扱うことはコボルに限らず ALGOL, PL/I, FORTRAN でも
そのように作れば可能である。

ALGOL

```

begin
  comment PROGRAM TEST OF KYUJO KEISAN;
  Boolean procedure B;
  B:=MBUKA<LBUKA;
  array E,T[1:7];
  integer I,NINZU,KSH,LBUKA,MBUKA,NO;
  real AE,HONPO,SHOKH,SAI,PKAZU,FUYO,ZANGYO,ZTIME,SHOTOK,ZEIKIN;
  integer procedure ZEI(S);
  real S;
  ZEI:=((if S<120000 then 0.2 else if S<100000 then 0.18 else
    if S<80000 then 0.15 else if S<50000 then 0.1 else
    if S<30000 then 0.05 else 0)*S-0.5;
  integer procedure FU(S,P);
  integer S,P;
  FU:=S*2000+P*1000;
  integer procedure ZAN(T,H);
  integer T; real H;
  ZAN:=H*0.005*T-0.5;
  procedure CLEAR;
  begin
    procedure A(I,S); integer N; string S;
    begin SPACE(N); PRINTS(S); end;
    for I:=1 step 1 until 7 do T[I]:=0;
    NINZU:=0; NEWPAGE;
    A(4,'BUKA'); A(7,'NO'); A(5,'HONPO'); A(3,'SHOKUMU');
    A(5,'FUYO'); A(5,'ZANGYO'); A(5,'SHOTOKU'); A(8,'ZEIKIN');
    A(7,'TEDORI'); CRLF(3);
  end;
  procedure TOTAL(A);
  accu A;
  begin
    if B then begin SPACE(4); PRINTS('TOTAL'); end
    else PRINTI(6,LBUKA);
    PRINTI(6,if B then NINZU else NO);
    for I:=1 step 1 until 7 do
      PRINTI(12,if I<2 then 7 else if I=3 then 6 else
        if I=4 then 8 else 10,A[I]);
    CRLF(2);
  end;
  CLEAR; KSH:=1;
  READIN;
  READ(LBUKA,NO,HONPO,SHOKH,SAI,PKAZU,ZTIME);
  if LBUKA=0 then begin TOTAL(T); go to END; end;
  if KSH=1 then begin KSH:=2; MBUKA=LBUKA; end;
  if B then begin TOTAL(T); CLEAR; MBUKA=LBUKA; end;
  ZANGYO:=ZAN(ZTIME,HONPO);
  FUYO:=FU(SAI,PKAZU);
  SHOTOK:=HONPO+SHOKH+FUYO+ZANGYO;
  ZEIKIN:=ZEI(SHOTOK);
  I:=1;
  for AE:=HONPO,SHOKH,ZANGYO,FUYO,SHOTOK,ZEIKIN,SHOTOK-ZEIKIN do
    begin E[I]:=AE; I:=I+1; end;
  for I:=1 step 1 until 7 do
    T[I]:=T[I]+E[I];
  NINZU:=NINZU+1;
  TOTAL(E);
  go to READIN;
END;
end;

```

PL/I

```

KYUYO: PROCEDURE OPTIONS(MAIN);
/* PROGRAM TEST OF KYUYO KEISAN */
DECLARE (T(7),AE,HONPO,SHOKUH,SAI,PKAZU,FUYO,ZANGYO,
        ZTIME,SHOTOK,ZEIKIN,TEDORI) FLOAT,
        (I,NINZU,KSH,INITIAL(1),LBUKA,HBUKA,NO) FIXED;
FU: PROCEDURE (S,P) FIXED;
    DECLARE (S,P) FLOAT;
    RETURN(S*2000+P*1000);
END FU;
ZEI: PROCEDURE (S) FIXED;
    DECLARE S FLOAT;
    (A(9),S) INITIAL (120000,100000,
        80000,50000,30000,0.2,0.18,0.15,0.1,0.05) FLOAT;
    DO I=1 TO 5;
        IF S>= A(I) THEN RETURN(S+R(I));
    END;
    RETURN(R);
END ZEI;
ZAN: PROCEDURE (T,H) FIXED;
    DECLARE (T,H) FLOAT;
    RETURN(H*0.005*T);
END ZAN;
CLEAR: PROCEDURE;
    T=0;
    PUT EDIT (PAGE,X(4),'BUKA',X(7),'NO',X(5),'HONPO',
        X(1),'SHOKUH',X(5),'FUYO',X(5),'ZANGYO',X(5),
        'SHOTOKU',X(8),'ZEIKIN',X(6),'TEDORI',SKIP(2));
END CLEAR;
TOTAL: PROCEDURE;
    PUT EDIT ('TOTAL',NINZU,T(1) DO I=1 TO 7)
        (X(3),A(5),F(9),(2)F(10),F(9),F(11),
        (3)F(13),SKIP(2));
END TOTAL;
CALL CLEAR;
READIN:
    GET LIST (LBUKA,NO,HONPO,SHOKUH,SAI,PKAZU,ZTIME);
    IF LBUKA=0 THEN
        DO; CALL TOTAL; GO TO LEND; END;
    IF KSH=1 THEN DO; KSH=2; HBUKA=LBUKA; END;
    IF LBUKA=HBUKA THEN
        DO; CALL TOTAL; CALL CLEAR; END;
    ZANGYO=ZAN(ZTIME,HONPO);
    FUYO=FU(SAI,PKAZU);
    SHOTOK=HONPO+SHOKUH+FUYO+ZANGYO;
    ZEIKIN=ZEI(SHOTOK);
    TEDORI=SHOTOK-ZEIKIN;
    BEGIN;
        DECLARE E(7) FLOAT;
        I=1;
        DO AE=HONPO,SHOKU,ZANGYO,FUYO,SHOTOK,ZEIKIN,TEDORI;
            E(I)=AE;
            I=I+1;
        END;
        T=T+E;
        NINZU=NINZU+1;
        PUT EDIT (LBUKA,NO) (SKIP(2),F(8),F(9));
        PUT EDIT (E(1) DO I=1 TO 7) ((2)F(10),F(9),F(11),
            (3)F(13));
        GO TO READIN;
    END;
LEND:;
END KYUYO;

```

FORTRAN

```

/JOE
/EXC      FTC
C        PROGRAM TEST OF KYUJO KEISAN
          FUNCTION IZFI(S)
          DIMENSION A(5),B(5)
          DATA A/120000,100000,80000,50000,30000/,
1          B/0.20,0.18,0.15,0.10,0.05/
          IZFI=0
          DO 7 I=1,5
            IF (S-A(I)) 7,8,8
          6 IZFI=S*B(I)
          GO TO 9
          7 CONTINUE
          8 RETURN
          END

/EXC      FTC
          SUBROUTINE CLEAR
          COMMON THONPO,TSHOKH,TZAN,TFUYO,
1          TSHOT,TZEI,TTEDO,ITNIN
          THONPO=0
          TSHOKH=0
          TZAN=0
          TFUYO=0
          TSHOT=0
          TZEI=0
          TTEDO=0
          ITNIN=0
          WRITE(6,1)
1          FORMAT(1H1,4X,4HROKA,7X,2HNO,5X,
2          5H1HONPO,3X,7HSHOKUMU,5X,4HFUYO,
3          5X,6HZANGYO,5X,7HSHOTOKU,8X,
4          6HZEIKIN,6X,6HTEDORI//)
          RETURN
          END

/EXC      FTC
          SUBROUTINE TOTAL
          COMMON THONPO,TSHOKH,TZAN,TFUYO,
1          TSHOT,TZEI,TTEDO,ITNIN,NINZU
          WRITE(6,12)NINZU,THONPO,TSHOKH,
2          TFUYO,TZAN,TSHOT,TZEI,TTEDO
12          FORMAT(1H0,3X,5HTOTAL,19,2F10.0,F9.0,F11.0,
3          3F13.0//)
          RETURN
          END

/EXC      FTC
          COMMON THONPO,TSHOKH,TZAN,TFUYO,TSHOT,
1          TZEI,TTEDO,ITNIN,NINZU
          IFU(S,P)=S*2000.0+P*1000.0
          IZAN(T,H)=H*0.005*T
          CALL CLEAR
          KSW=1
11          READ(5,2)LRUKA,NO,HONPO,SHOKH,SAI,PKAZU,ZTIME
2          FORMAT(16,14,F6.0,4F5.0)
          IF(LRUKA.EQ.0)GO 10 9
          GO TO (3,4),KSW
          4 IF(LRUKA-MHUKA) 5,0,5
          3 KSW=2
          LRUKA=LRUKA
          ZANGYO=IZAN(ZTIME,HONPO)
          FUYO=IFU(SAI,PKAZU)
          SHOTOK=HONPO+SHOKH+FUYO+ZANGYO
          ZEIKIN=IZFI(SHOTOK)
          TEDORI=SHOTOK-ZEIKIN
          THONPO=THONPO+HONPO
          TSHOKH=TSHOKH+SHOKMU
          TZAN=TZAN+ZANGYO
          TFUYO=TFUYO+FUYO
          TZEI=TZEI+ZEIKIN
          TTEDO=TTEDO+TEDORI
          NINZU=NINZU+1
          WRITE(6,10)LRUKA,NO,HONPO,SHOKMU,FUYO,
2          ZANGYO,SHOTOK,ZEIKIN,TEDORI
10          FORMAT(1H0,18,19,2F10.0,F9.0,F11.0,3F13.0)
          GO TO 11
          5 CALL TOTAL
          CALL CLEAR
          GO TO (6,15),KEND
          6 KEND=2
          GO TO 5
15          STOP
          END

```

COBOL

```

IDENTIFICATION DIVISION.
PROGRAM-ID. KYUDO-KEISAN.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. TOSBAC-3400-D.
OBJECT-COMPUTER. TOSBAC-3400-D.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CARD-FILE ASSIGN TO IN.
    SELECT PRINT-FILE ASSIGN TO OUT.

DATA DIVISION.
FILE SECTION.
FD CARD-FILE
    DATA RECORD IS CARD-REC.
01 CARD-REC.
    02 LRUKA PICTURE 9(6).
    02 NA PICTURE 9(5).
    02 HONPO PICTURE 9(6).
    02 SHOKMU PICTURE 9(5).
    02 SAI PICTURE 9.
    02 PKAZU PICTURE 9(2).
    02 ZTIME PICTURE 9(3).
    02 FILLER PICTURE X(52).
FD PRINT-FILE
    DATA RECORD IS PRINT-REC.
01 PRINT-REC PICTURE X(119).
WORKING-STORAGE SECTION.
77 MRUKA PICTURE 9(10) VALUE ZERO.
77 WORK-3 PICTURE 9(199).
77 I PICTURE 9(10).
77 J PICTURE 9(10).
01 MY-1.
    02 TOTAL-WORK PICTURE 9(10) OCCURS 8 TIMES.
01 MY-2.
    02 WORK-AREA PICTURE 9(10) OCCURS 8 TIMES.
01 HEADER.
    02 FILLER PICTURE X(191) VALUE #1 RUKA NO HON
    - "PO SHOKUMU" FUYO ZANGYO SHOTOKU ZEIKIN
    - "TEDORI".
01 PRINT-LINE.
    02 FILLER PICTURE X VALUE #0.
    02 LP PICTURE Z(10) OCCURS 9 TIMES.
01 TOTAL.
    02 FILLER PICTURE X(11) VALUE #0 TOTAL*.
    02 T PICTURE Z(10) OCCURS 8 TIMES.

PROCEDURE DIVISION.
OPEN INPUT CARD-FILE.
OUTPUT PRINT-FILE.
PERFORM CLEAR.
YONIKOME.
    READ CARD-FILE AT END GO TO TOTAL-PRINT.
    IF MRUKA IS EQUAL TO ZERO MOVE LRUKA TO MRUKA.
    IF MRUKA IS EQUAL TO LRUKA GO TO KEISAN.
    MOVE LRUKA TO MRUKA.
    PERFORM TOTAL-PRINT THRU CLEAR.
KEISAN.
    MOVE LRUKA TO LP (1).
    MOVE NA TO LP (2).
    MOVE HONPO TO LP (3).
    MOVE SHOKMU TO LP (4).
    COMPUTE WORK-AREA (4) = HONPO * 0.005 * ZTIME.
    COMPUTE WORK-AREA (5) = SAI * 2000 * PKAZU * 1000.
    ADD HONPO SHOKMU WORK-AREA (4) WORK-AREA (5)
    GIVING WORK-AREA (6).
    MOVE ZERO TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 30000 MOVE 0.09 TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 50000 MOVE 0.10 TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 80000 MOVE 0.15 TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 100000 MOVE 0.18 TO WORK-3.
    IF WORK-AREA (6) IS NOT LESS THAN 120000 MOVE 0.20 TO WORK-3.
    MULTIPLY WORK-AREA (6) BY WORK-3 GIVING WORK-AREA (7).
    SUBTRACT WORK-AREA (7) FROM WORK-AREA (6)
    GIVING WORK-AREA (8).
    ADD HONPO TO TOTAL-WORK (2).
    ADD SHOKMU TO TOTAL-WORK (3).
    ADD 1 TO TOTAL-WORK (1).
    PERFORM LP-MOVE VARYING I FROM 4 BY 1 UNTIL
        I GREATER 8.
    WRITE PRINT-REC FROM PRINT-LINE.
    GO TO YONIKOME.
TOTAL-PRINT.
    PERFORM TOTAL-PRINT.
    GO TO DHARI.
TOTAL-PRINT.
    PERFORM TOTAL-MOVE VARYING I FROM 1 BY 1 UNTIL
        I GREATER 8.
    WRITE PRINT-REC FROM TOTAL.
CLEAR.
    PERFORM TOTAL-CLEAR VARYING I FROM 1 BY 1 UNTIL
        I GREATER 8.
    WRITE PRINT-REC FROM HEADER.
TOTAL-CLEAR.
    MOVE ZERO TO TOTAL-WORK (1).
TOTAL-MOVE.
    MOVE TOTAL-WORK (1) TO T (1).
LP-MOVE.
    ADD 1, I GIVING J.
    MOVE WORK-AREA (1) TO LP (J).
    ADD WORK-AREA (1) TO I (1).
DHARI.
    CLOSE CARD-FILE,PRINT-FILE.
    STOP RUN.

```

ICBEND

§ 3 自動図化とプログラム例

電子計算によって設計から図化まで作業を行なった標準設計 P C T も 8 月発刊されるがこの P C T を例にとって自動図化の概要をのべてみたい

既刊の標準設計は手書きによる図面であったため

- (1) 審査するのに作図と同じ程度の日時を要する。
- (2) 審査するにも内容の理解できる技術者が当る必要がある。
- (3) 多量の図面となると製図者が一人でないため字型、線の太さが非常にまちまちである。

等の弊害があった。また、P C 橋の場合どんなに熟練された技術者でも設計から図面の仕上がりまで 2 週間必要である。

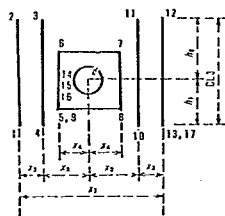
しかし、自動設計によれば 1 時間 30 分もあれば設計から図面まで仕上がり、自動図化によった場合代表的な何枚かを精査しておけば、後は何枚書かせようが転記ミスも脱落もなく、線の太さ、濃さ、字型が一定で統制のとれた図面となる。

プログラムはインプットデータが変わった場合でもその対象となる構造物が同一のプログラムで処理できるように汎用性を持っている事が必要であり、斜角のある場合でも左(L)、右(R)を+、-で指示することによって描けるようにもしたが、普通行なっている鉄筋コンクリート構造物の設計方式では配力鉄筋や用心鉄筋のように計算で求める事なく設計者の判断によって決められるような要素が非常に多く含まれておりプログラミングにとってはやっかいな事、また図化機のペン先を上げて動く距離を最少にしないと作図時間が長くなるので注意が必要である等一考を要する点であった。P C 橋の設計図面は最低一般構造図と主げた構造図が必要であるが、当室の行なった自動図化でも計算機が改造前であるためプログラムを主げた構造図、一般構造図の 2 つに分け図面相互間の連絡は対照番号でわかるようにした。

次図は標準設計 P C T の一部であるが、計算時間は支間、本線幅、歩道幅、橋格を与え、それを用いた演算の実行に 20 秒、その結果に基づいて図化機をコントロールする指令を作り出すのに 5 分、この結果により図化機が 1 組 2 枚の図面を書き終るまでに一般構造図に 30 分、主げた構造図に 50 分かかっている。(CDC-3600 による)

次に国鉄構造物設計事務所で作製された F O R T R A N の図化プログラムと
土木研究所の A L G O L による図化プログラムを比較してみる。

この手続は I 型 P C げた設計図の一般図中において側面図の部分に鋼棒の取
付位置を描くのに用いるものである。図面の左下端から作図が始まり作図が終
つたときは右下端に作図ペンがくるようになっている。



$$h_2 = \frac{CL3}{2} - 10$$

$$h_1 = CL3 - h_2$$

BOM	X ₁	X ₂	X ₃	X ₄	d
24	180	70	20	60	35
27	195	775	20	675	35

まず国鉄と土木研究所における図化プログラムの標準手続きは次のとおりで
ある。

- (1) 国鉄の F O R T R A N による図化用サブルーチン

CALL DRFIL (0)

入力紙テープを 100 フィート送り出し、製図機の No. 1 のペンを選択し、描
画スピードの制御は、制御卓の No. 1 のダイヤルを用いることを指示する。通常、
製図プログラムが一番始めに用いられるステートメントである。

CALL DRFIL (1)

製図ペンを上げることを指示する。

CALL DRFIL (2)

製図ペンを下げのことを指示する。

CALL DRFIL (3, PS)

使用する製図ペンを選択する。PSの位置に1が書かれていると、 $\phi 1$ のペンが選ばれることになる。

CALL DRFIL (4, SF)

自動製図機操作卓の描画速度制御ダイヤルを選択する。SFの位置に1が書かれると、 $\phi 1$ のダイヤルを使用して描画速度を制御できる。

CALL DRFIL (5, XSC, YSC)

設計図面の縮尺を指定するのに用いる。XSCの位置にはX軸の縮尺、YSCにはY軸の縮尺を記入する。

CALL DRFIL (6, KOUNT, XAR, YAR, PEN)

作図ペンを移動させるのに用いる。KOUNTの位置には、ペンを移動して結んでいく点の数を記入する。XAR, YARには、結んでいく各点のX座標値とY座標値が貯えてあるアレー名を書く。PENの位置に1が書かれると、作図ペンは上ったまま移動し、2が書かれると下がった状態で移動する。

CALL DRFIL (7, N, XS, YS, XE, YE)

円弧を描く場合に用いる。XS, YSには描くべき円弧の始点のX座標値、Y座標値が入り、XE, YEには終点のX座標値、Y座標値が入る。Nの位置に1を記入すると、作図ペンは始点から終点に時計方向に円弧を描き、2を記入すると反時計方向に円弧を描く。

CALL DRFIL (13, KOUNT, XAR, YAR)

破線を描く場合に用いる。KOUNTの位置には、破線で結ぶ点の数を記入し、XAR, YARには、各点のX座標値、Y座標値が貯えてあるアレー名を記入する。

(2) 土木研究所のALGOLによる図化手続き

PREADY (X_2, Y_2); 今のペンの位置が (x_2, y_2) となるよう座標系を設定せよ。

PEND; ペンを原点にもどせ。

PMOVE (X_2, Y_2); ペンを上げて (x_2, y_2) に移動せよ。

PSENX (X_2, Y_2); 今のペンの位置より (x_2, y_2) まで直線を引け

PENX (X_2, Y_2, R); 今のペンの位置より (x_2, y_2) まで半径Rで円弧

を描け。ただしRが正なら右回り負なら左回りとする。

PSENA (A, L) ; 今のベンの位置よりX方向を0度とした角度Aで長さLミリの直線を描け。

POUTI (A, S, N, E) ; 式Eの値を方向角A, 字の大きさSミリ有効けた数Nの整数タイプで描け。

POUTR (A, S, N, E) ; 式Eの値を同じく実数タイプ(10を含む)で描け。

POUTF (A, S, n₁, n₂, E) ; 式Eの値を同じく小数点以上n₁けた以下n₂けたの形で描け。

POUTS (N, A, S, '.....') ; ...という文字をN文字分、方向角A、文字の大きさSで描け。

(3) 国鉄の鉄筋配置図化用ブルーチン

また手続き名とパラメータは次のとおりである。

CALL BAFIX (BDM, CL3, S)

パターンを描くのに必要な入力データは、使用する鋼棒の太さBDM、座金の縦方向の寸法CL3、パターンの縮尺Sの3種類である。これらの入力データは、パラメータとしてサブルーチンに与えられ、メイン・プログラムおよび他ルーチンに、サブルーチンと連結されている。

(4) 土木研究所の鉄筋配置図化用手続

procedure BARSET(X, Y, DIA, DIS, S, N) ;

X, Y : 鋼棒の中心座標

DIA : 鋼棒の太さ

DIS : 座金の縦方向の寸法

S : 図面の縮尺

N : 図面の大きさ

```

procedure BARSET(X,Y,DIA,DIS,S,N);
  real X,Y,DIA,DIS;
  integer S,N;
  begin real X1,X2,X3,X4,Y1,Y2,Y3;
    integer I;
    real procedure D(X);
      real X;
      D:=X/S#N;
    Boolean B;
    B:=DIA#24;
    X1:=if B then D(90) else D(97.5);
    X2:=if B then D(70) else D(77.5);
    X3:=if B then D(60) else D(67.5);
    Y3:=X3;
    Y4:=X+D(35);
    Y1:=D(DIS/2-10); Y2:=D(DIS)-Y;
    Y1:=Y+Y1; Y2:=Y-Y2;
    for I=1,2 do
      begin
        X1:=if I=1 then Y-X1 else X+X2;
        X2:=if I=1 then X-X2 else Y+Y2;
        PMOVE(X1,Y2); PSENK(X1,Y1);
        PMOVE(X2,Y1); PSENK(X2,Y2);
      end;
      X3:=X-X3; X2:=Y+Y3;
      Y3:=Y-Y3; Y2:=Y+Y3;
      PMOVE(X3,Y3); PSENK(X3,Y2);
      PSENK(X2,Y2); PSENK(X2,Y3);
      PSENK(X3,Y3);
      PMOVE(X4,Y); PSENK(X4,Y,D(35));
      WSCALE(X,D(DIS/2)-30,0,3,3,S);
      PMOVE(X,Y);
    end;

```

線分1-2, 8-4, 10-11,
12-13を描く

.....矩形5 6 7 8 9を描く

.....半径dの円を描く

.....図下へ縮尺を書く

```

SUBROUTINE DPL1(XPRN,CL2,5)
DIMENSION X(17),Y(17)
IF (TOL>24) 50,24,27
24 X(1)=0.
   X(2)=0.
   X(3)=20.
   X(4)=20.
   X(5)=0.
   X(6)=20.
   X(7)=150.
   X(8)=150.
   X(9)=X(5)
   X(10)=100.
   X(11)=100.
   X(12)=180.
   X(13)=107.5
   X(14)=107.5
   X(15)=107.5
   X(16)=107.5
   X(17)=180.
   Y(1)=0.
   Y(2)=CL2
   Y(3)=CL2
   Y(4)=0.
   Y(5)=CL2/2.-70
   Y(6)=CL2/2.+70
   Y(7)=Y(4)
   Y(8)=Y(5)
   Y(9)=Y(5)
   Y(10)=0.
   Y(11)=CL2
   Y(12)=CL2
   Y(13)=0.
   Y(14)=CL2/2.-10
   Y(15)=Y(10)
   Y(16)=Y(15)
   Y(17)=0.
GO TO 100
27 X(1)=0.
   X(2)=0.
   X(3)=20.
   X(4)=20.
   X(5)=0.
   X(6)=30.
   X(7)=105.
   X(8)=145.
   X(9)=2(5)
   X(10)=175.
   X(11)=175.
   X(12)=195.
   X(13)=195.
   X(14)=115.
   X(15)=115.
   X(16)=115.
   X(17)=195.
   Y(1)=0.
   Y(2)=CL2
   Y(3)=CL2
   Y(4)=0.
   Y(5)=CL2/2.-77.5
   Y(6)=CL2/2.+77.5
   Y(7)=Y(4)
   Y(8)=Y(5)
   Y(9)=Y(5)
   Y(10)=0.
   Y(11)=CL2
   Y(12)=CL2
   Y(13)=0.
   Y(14)=CL2/2.-10
   Y(15)=Y(14)
   Y(16)=Y(15)
   Y(17)=0.
100 SS=0.0145
   CALL DPL1(5,SS,SS) -----線分の指定
   DO 15 I=1,17
     X(I)=X(I)*SS
   15 Y(I)=Y(I)*SS
   CALL DPL1(2,2) -----型図への指定
   DO 10 I=1,2
     CALL DPL1(2,2,X(I),Y(I),2)
   10 CALL DPL1(6,2,X(1),Y(1),1) } 部分1-2, 3-4を描く
     DO 1=5,9
       CALL DPL1(6,5,X(1),Y(1),2) } 型図5,6,7,8を描く
     DO 20 I=9,11,2
       CALL DPL1(6,2,X(I),Y(I),1)
   20 CALL DPL1(6,2,X(11),Y(11),2) } 部分10-11, 12-13を描く
     CALL DPL1(6,2,X(15),Y(15),Y(15),Y(15))
     CALL DPL1(7,2,X(15),Y(15),X(15),Y(15)) 中心の円を描く
     CALL DPL1(6,2,X(12),Y(12),1) } 作図ペンを右下隅に移動する
50 CALL DPL1(1)
RETURN
END

```

鋼棒の太さ 24 mm の場合の
 各点の X, Y 座標

鋼棒の太さ 27 mm の場合の
 各点の X, Y 座標

§ 4. 電算室よりのお知らせ

〔1〕 電子計算機移動表

本所電子計算機稼働表

自昭和43年4月

至昭和44年3月

年 月	通電時間 (分)	check 時 間 (分)	down 時 間 (分)	使用時間 (分)	idle 時 間 (分)	そ の 他 (分)
43・4	13840	280	830	1549	895	10286
5	20700	840	15	4024	3523	12298
6	19685	1460	1800	5608	2521	8296
7	21265	1705	645	8432	1563	8922
8	32128	2294	1318	5931	1008	21577
9	24154	1655	528	7624	1872	12475
10	30540	1537	340	14800	2689	11174
11	28560	1499	27	18869	1790	6375
12	34670	1211	942	16175	958	15384
44・1	24150	1846	440	12946	599	8219
2	28770	1626	80	13064	2395	11605
3	28437	1572	111	16124	1988	9142
合 計	306899	17525	7076	125146	21801	135753
平 均	25575	1460	590	10429	1817	11313

千葉支所電子計算機稼働表

自昭和43年4月

至昭和44年3月

年 月	通電時間 (分)	check 時 間 (分)	Down 時 間 (分)	使用時間 (分)	idle 時 間 (分)	そ の 他
43・4	16420	1000	680	5060	9680	
5	18210	1110	420	5850	10830	
6	17030	880	0	5650	10550	
7	19400	1040	1220	6260	9880	
8	19840	720	420	6940	11760	
9	13940	840	420	7630	5050	
10	21430	770	0	15510	5150	
11	28970	780	1730	18140	9320	
12	28910	720	2250	13710	12230	
44・1	19140	740	990	12280	5130	
2	24720	3540	0	18750	2430	
3	26640	2265	150	13150	8075	
合 計	254650	14355	8280	128930	98585	
平 均	21221	1196	690	10744	8215	

本 所 電 子 計 算 機 の 利 用 時 間

自 昭 和 4 3 年 4 月

至 昭 和 4 4 年 3 月

年 月	日 数	件 数	月総時間	1日平均	河 川 系		道 路 系		そ の 他	
			(分)	時間(分)	件 数	時間(分)	件 数	時間(分)	件 数	時間(分)
43.4	9	42	1,549	172	18	591	18	684	6	274
5	26	261	4,024	155	61	1,465	127	1,207	73	1,352
6	23	586	5,608	244	228	2,598	272	2,646	86	364
7	26	601	8,432	324	139	2,477	407	5,214	55	741
8	27	455	5,931	220	86	1,545	276	3,688	93	698
9	24	579	7,624	318	165	2,700	253	2,401	161	2,523
10	28	762	14,800	529	242	4,014	291	7,160	229	3,626
11	25	766	18,869	755	145	3,244	245	6,956	376	8,669
12	25	430	16,175	647	121	3,249	152	4,400	157	8,526
44.1	22	485	12,946	588	179	3,899	188	3,360	118	5,687
2	23	528	13,064	568	191	4,158	224	4,711	113	4,195
3	25	471	16,124	645	159	4,858	174	3,596	138	7,677
合 計	283	5,966	125,146	5,165	1,734	34,798	2,627	46,023	1,605	44,332
平 均	24	497	10,429	430	145	2,900	219	3,835	134	3,694

〔 2 〕 プログラム名簿

プログラム登録人員総括表

所 属		登録人員 (A)	登録人員 (B)	計
土木研究所本所	A	1 0 6		1 0 6
千 葉 支 所	B	1 7 2		1 7 2
赤 羽 支 所	C	1 0 1		1 0 1
篠崎試験所	D	2 1		2 1
鹿島試験所	E	3		3
新潟試験所	F	4		4
土木研究所小計		4 0 7		4 0 7
本 省	G	1 9 0	6 4	2 5 4
東 北 地 建	K	8 9	2 5 8	3 4 7
北 陸 地 建	L	1 2 7	5 9 2	7 1 9
関 東 地 建	M	2 6 7	9 5	3 6 2
中 部 地 建	N	1 2 5	2 4 5	3 7 0
近 畿 地 建	P	9 8	2 5 1	3 4 9
中 国 地 建	Q	1 0 2	9 4	1 9 6
四 国 地 建	R	7 8	1 6 4	2 4 2
九 州 地 建	S	7 8	1 5 5	2 3 3
地 建 小 計		9 6 4	1 8 5 4	2 8 1 8
都府県関係	T	3 4 8		3 4 8
公団・公社	U	6 0		6 0
そ の 他	V	1 9 2		1 9 2
小 計		6 0 0		6 0 0
合 計		2,1 6 1	1 9 1 8	4 0 7 9

A : 土木研究所にて研修

B : 各地建等にて研修

[3] TOPS - 14 T - ALGOL ERROR TABLE

1) コンパイル時に検出された誤りの表示

G-XXX LINE YYYYY Z Z Z Z

XXX エラー番号

YYYY そのエラーを検出したソース・プログラム上の行数
エラー番号が200番台以外では印字されない。

Z Z Z ローマ字で書かれエラーの内容と処理を示す。

2) 実行時に検出された誤りの表示

G-XXX *****

の形で誤りが検出された時点で印字

3) コンパイル時に検出される誤り

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
101	7	文の初めに名前があり次に区切り記号がない。	NAMAE(IDENTIFIER) NO TUGI NI KUGIRIKIGO GA NAI. TUGI NO 'END', 'ELSE', ';' NO MAE MADE MUSI SURU	begin, end, else の前まで 無視	ALGOL:= ABCD+B;
102	7	文の初めが名前や区切り記号でない	BUN(STATEMENT) NO HAZIME NI NAMAE(IDENTIFIER), KUGIRIKIGO (DFLIMITER) GA NAI.	"	123:=A+B;
103	7	文の初めに許されない綴り区切り記号がある	BUN(STATEMENT) NO HAZIME NO KUGIRIKIGO(SPELLED DELIMITER) GA MAHIGATTE IRU.	"	step 1 untile Ndo
104	5	go to ステートメントの最後に end, else が来ない	TOBIKOSIBUN(GOTO STATEMENT) NO OWARI GA MATIGATTE IRU. ';' O OGINATTA.	;を追加	if ALGOL =1 begin
105	5	if ステートメントで then が 来ない	ZYOKENSETU(IF CLAUS) NI 'THEN' GA NAI. 'THEN' O OGINATTA.	then を追加	for ALGOL begin
106	5	for ステートメントで、又は do が来ない	KURIKAESI YOSO(FOR LIST ELE- MENT) NO OWARI GA MAHIGATTE IRU. 'DO' O OGINATTA.	do を追加	for ALGOL= =1 step 1 untile N begin

エラー 番 号	grado	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
107	7	if と then の数が合わない if < then	'IF' TO 'THEN' NO KAZU GA AWANAI. TUGI NO 'END', 'ELSE', ';' NO MAE MADE MUSI SURU,	begin,end,else ;の前まで無視	
108	7	then と else の数が合わない then < else	'ELSE' GA OOSUGIRU. TUGI NO 'END', 'ELSE', ';' NOMAE MADE MUSI SURU.	//	
109	7	式の後に:=が来た。	SAHEN(LEFT PART) NI SIKI(EXPRESSION) GA ARU. TUGI NO 'END', 'ELSE', ';' NO MAE MADE MUSI SURU.	//	ALGOL:= A+B:=10;
110	9	end が少ない	'END' GA SUKUNAI. 'COMPILE' O YAMERU.	stop	
111	5	vec と tor 又は(と)又は [と]又は if と then と else の数が合わない。右の方が少ない。	'VEC' TO 'TOR', '(' TO ')', '[' TO ']', 'IF' TO 'THEN' TO 'ELSE' NO KAZU GA AWANAI	tor,),]を追 加 if の場合は begin,end,;前 まで無視	A:=(ALGOL+ B)/((C+D)+ F);
112	7	(と)又は[と]の数が合わない。 (<)又は[<]	(' TO ')', '[' TO ']' NO KAZU GA AWANAI. TUGI NO 'END', 'ELSE', ';' NO MAE MADE MUSI SURU	begin,end,else, ;の前まで無視	A:=(A+B)/ (B+C);

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
113	7	vec と tro の数が合わない。 vec < tor	'VEC' TO 'TOR' NO KAZU GA AWA- NAI. TUGI NO 'END', 'ELSE', ' ; ' NO MAE MADE MUSI SURU.	//	initial vec, tro tro
114	5	型の宣言詞の次に名前又は配列 の宣言詞又は続きの宣言詞が来 ない。	KATA NO NARABI(TYPE LIST) GA MATIGATTE IRU. TUGI NO ' ; ' NO MAE MADE MUSI SURU	; の前まで又は' 又は initial..., まで無視	real 'ALGOL';
115	5	型の並び又は配列の並びで区切 り記号がない。	KATA NO NARABI(TYPE LIST) MA- TAWA HAIRETU NO NARABI(ARRAY LIST) NO KUGIRI GA MAHIGATTE IRU.	その element を 無視し、追加	real ALGOL true
116	5	上下限の並びに ; がない。	ZYOKAGEN NO NARABI(BOUND PAIR LIST) NI ' ; ' GA NAI. 'COMPIL' SONO MAMA TUZUKERU.	なにもしない	array A[1,2];
117	5	上下限の並びの終りが ' ,] , ' ; 以外である。	ZYOKAGEN NO NARABI(BOUND PAIR LIST) OWARI GA MATIGATTE IRU. '] ' O OG NATTA.	] を追加する	array A[1:2 begin
118	5	上下限に ; が2つ以上ある。	ZYOKAGEN(BOUND PAIR) NO NAKA NI ' ; ' GA 2 KAI ARAWARETE IRU. ' , ' NI KAETA	, に変更	array A[1:2;];

エラー 番 号	grade	原 因	エ ラ ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
119	5	配列の並びの終りがまちがひ。	HAIRETU NO NARABI(BOUND PAIR LIST) NO OWARI GA MATIGATTE IRU. ' ' O OGINATTA	, を追加する	array A[1:2] 1 2 3
120	5	配列の並び又は型の並びの区切り記号がまちがっている。許されるもの ; , ' and, initial	KATA NO NARABI(TYPE RIST) MATAWA HAILETU NO—NARABI(ARRAY LIST) NO OWARI GA MATIGATTE IRU.	; を追加する	real A * B ;
121	5	初期値, 設定の部の最後に許されない区切り記号がある 許されるもの , ; , end	' INITIAL' NO ATO NO SIKI(EXPRESSION) NO OWARI GA MATIGATTE IRU. ' ; ' O OGINATTA.	; を追加する	initial.....;B
122	5	own の次が型でない, 又は array でない	' own' NO TUGI GA KATA(TYPE) MATAWA 'ARRAY' DE NAI. 'REAL' O OGINATTA.	real を追加	own ALGOL
123	5	external の次が名前又は区切り記号以外である。	' EXTERNAL' NO ATO NO NAMA E (IDENTIFIER) NO NARABI NI MATIGAI GA ARU. SORLE O MUSI SURU.	その element を無視する	external 1 2 3 ;
124	5	external で名前の後に区切り記号がない又は ' , end, 以外である。	' EXTERNAL' NO ATO NO NAMA E (IDENTIFIER) NO NARABI NO OWARI GA MATIGATTE IRU. ' , ' O OGINATTA	, を追加	external A*B ;

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
125	5	external で名前の前が区切り 記号でないxでない場合	'EXTERNAL' NO ATO NO NAMA (IDENTIFIER) NO NARABI NI MA- TIGAI GA ARU. '*' TO MINASI- TA.	*と判断する	external A, B, C;
126	5	switch の後が名前でない	'SWITCH' MEI (SWITCH IDENTIFI- ER) GA NAI. 'COMPILE' SONO MAMA TUZUKERU.	その element を 無視する	switch 1 2 3
127	5	switch に := がない	'SWITCH' MEI (SWITCH IDENTIFI- ER) NO ATO NI ':' '=' GA NAI. ' := ' O OGINATTA.	:= を追加	switch A, B, C;
128	5	switch の終りの区切り記号が まちがっている。	'SWITCH' NO SENGEN (SWITCH DE- CLARATION) NO OWARI GA MATI- GATTE IRU. ';' O OGINATTA	; を追加	switch A := L ₁ , L ₂ ,
129	5	procedure の次が名前でない	TETUZUKIMEI (PROCEDURE IDEN- TIFIER) GA NAI 'COMPILE' SO- NO MAMA TUZUKERU.	そのエレメントを 無視	procedure 1 2 3 (
130	5	手続き名の後に(, (または; がない	TETUZUKIMEI (PROCEDURE IDENTI- FIER) NO ATO NI '(', ';' GA NAI. '(' O OGINATTA.	; を追加	procedure ID*

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
131	5	手続きで[, (の次に名前が来ない。	KARIPARAMETA(FORMAL PARAMETER) GA NAMA(E IDENTIFIER) DE NAI. SORE O MUSI SITA.	その element を 無視	procedure ID(123)
132	5	仮パラメタの並びの区切り記号がない, 又は)又は]	KARIPARAMETA NO NARABI(FORMAL PARAHETER LIST) NO KUGIRI GA MATIGATTE IRU. ', ' O OGINATTA	, を追加	procedure ID (ID*ID)
133	5	仮パラメタ部の後に; が来ない	KARIPARAMETABU(FORMAL PARAME- TER PART) NO ATO NI ', ' GA NAI. , ' O OGINATTA	; を追加	P ID(...)*
134	5	規制部がまちがつている	KISEIBU(SPECIFICATION PART) NO NAMA(E NO NARABI IDENTIFIER LI- ST) GA MATIGATTE IRU	, 又は; まで無 視	
135	5	仮パラメタ部にない名前を規制した	KARIPARAMETABU(FORMAL PARAME- TER PART) NI NAI NAMA(E IDENTI- FIER) GA KISEI SARE TE IRU.	"	
136	6	規制詞がまちがつている	MATIGATTA KISEISI(SPECIFIER) GA MATIGATTE IRU. TUGI NO ', ' MADE MUSI SURU.	; の前まで無視	

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ 1 10 20 30 40 50	処 理	例
137	6	2重規制	ONAZI NAMAЕ(IDENTIFIER) GA 2 KAI IZYO KISEI SARETE IRU. TUGI NO ',',';' MADE MUSI SURU	;の前まで無視 又は,まで無視	
138	8	identifier table と declara- lion table オ ー バ ー	NAMAE(IDENTIFIER), TEISU(CONS- TANT) YA KIGORETU(STERING) GA OOSUGIRU. 'TABLE' GA AHURETA.	programの最後 まで無視	
139	9	Machine error	'MACHINE ERROR'	STOP	
140	3	endの後 letter,digit,',', end comment 以外のものが来た	'END' NO ATO NO KIHONKIGO GA MATIGATTE IRU. MUSI SITA.	そのelement 無 視	end begin
141	3	基本記号にないものが来た	KIHONKIGO DE NAI KIGO GA ARU. MUSI SITA.	そのelement 無視	¥
142	3	1つの数の中に・が2回以上又 は:の後に・が来た。	KAZU(NUNBER) NO NAKA NI ', ' GA 2 KAI IZYOO ARAWARETE IRU. MA- TAWA '10' NO ATO NI ', ' GA AR ARAWARETE IRU	"	3.14.54 3.010 3.0
143	3	数の中に:が2回以上使われ ている	KAZU(NUMBER) NO NAKA NI '10' GA 2 KAI IZYOO ARAWARETE IRU. MUSI SITA	"	3.010 110

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
144	3	数の中にdが2回以上使われている	KAZU(NUMBER) NO NAKA NI 'D' GA 2 KAI IZYOO ARAWARETE IRU. MUSI SITA.	#	3.0 d d
145	3	数の中にiが #	KAZU(NUMBER) NO NAKA NI 'I' GA 2 KAI IZYOO ARAWARETE IRU. MUSI SITA.	#	3.0 i i
146	3	数の中でi又はdが使われた後に続いてdi又はidが使われた	KAZU(NUMBER) NO_NAKA NI 'D', 'I', 'DI', 'ID', GA 2 KAI IZYOO ARAWARETE. MUSI SITA.	#	
147	3	)...で:が来ない	)...:(DE ':' GA NAI. MUSI SITA	#	)...(
148	3	)...:で(が来ない	)...:(DE '(' GA NAI. MUSI SITA.	#	)...:begin
149	3	綴り区切り記号のまちがい	TUZURI KUGIRIKIGOO(SPELLED DELIMITER) NO TUZURI GA MATI- I ETTAGAITTE IRU MUSI SITA.	#	begin
150	3	基本記号にないものが来た	KIHONKIGOO DE NAI KIGO GA ARU. MUSI SITA.	#	¥

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
151	3	T-3400 にない character が 入つて来た	YURUSARENAI KIGO GA ARU. MUSI SITA	#	
201	9	ブロックが大きすぎてスタック があふれた	'BLOK' GA OOKI SUGIRU. 'STACK' GA AHURETA.	コンパイル打ち切り	
202	9	ブロックが大きすぎて TREE テーブルがあふれた	'BLOCK' GA OOKI SUGIRU. 'TREE TABLE' GA AHURETA	コンパイル打ち切り	
203	9	定数や記号列が多すぎて、コン スタント・テーブルがあふれた	TEISU(CONSTANT) YA KIGORETU (STRING) GA OOSUGIRU. 'CONS- TANT TABLE' GA AHURETA	コンパイル打ち切り	
204	9	一つのブロックで使っている変 数が多すぎる	'BLOCK' DE TUKATTE IRU NAMA (IDENTIFIER) GA OOSUGIRU	コンパイル打ち切り	
211	2	宣言されていない名前がある	NAMAE(IDENTIFIER) GA SENGEN SARETE INAI. ZISSUGATA-REAL TYPE) TO MINASITA	実数型変数とみな す	
212	4	名前が続いている (区切り記号がない)	NAMAE(IDENTIFIER) GA TUZUITE IRU. MUSI SITA.	その名前を無視す る	real . A, B ₍₁₎ C ;

エラー 番号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
213	7	配列の宣言で、上下限の並びが くるまえに；がでてきた	';' WA MATIGAI. KUUBUN(DUMMY STATEMENT) TO MINASITA.	その宣言文を無 視する	array A,B;
214	4	区切り記号が続いている	KUGIRIKIGO(DELIMITER) GA TUZU- ITE IRU. MUSI SITA.	その区切り記号 を無視する	real A,B,,C;
215	7	上下限の並びがまちがつている	JOKAGEN NO NARABI (BOUND PAIR LIST) GA MATIGATTE IRU. KUU- BUN(DUMMY STATEMENT) TO MINA- SITA.	その宣言文を無 視する	array A[1:];
217	5	上下限の並びの終りに]がない	';' WA MATIGAI. ']' O OGINA- TTA.	]をおぎなう	array A[1:10;
218	5	手続き名の後に(または；がな い	TETUZUKIMEI(PROCEDURE IDENTI- FIER) NO ATONI '(' MATAWA ';' GA NAI. '(' MATAWA ';' O O- GINATTA.	(または；をお ぎなう	procedure P,A);
219	5	(の後に名前がない	KARIPARAMETA(FORMAL PARAMETER) GA NAI. MUSI SURU.	その element を 無視する	procedure P(123,

エラー 番号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
220	5	, がない (区切り記号のまちがい)	KARIPARAMETABU(FORMAL PARAME- TER PART) NO KUGIRIKIGO(DELIM- ITER) GA MATIGATTE IRU. ', ' TO MINASITA	, とみなす	procedure P(A*B);
221	5	) のあとに ; がない (名前が続いている)	KARIPARAMETABU(FORMAL PARAME- TER PART) NO ATONI ', ' GA NAI. ';' O OGINATTA	; をおきなう	procedure P(A) B
222	5	) のあとに ; がない (区切り記号がまちがっている)	KARIPARAMETABU(FORMAL PARAME- TER PART) NO ATONI ', ' GA NAI. ',' TO MINASITA	, とみなす	procedure P(A)
223	5	値とりの仮パラメタにスイッチ 名手続き名または記号列がある	ATAI NO BU(VALUE PART) NO SI- TEI GA MATIGATTE IRU. 'VALUE' NO SITEI O MUSI SITA;	value の指定を 無視する	
224	4	switch に := がない	'SWITCH' NO SENGEN(SWITCH DEC- LARATION) DE ':= ' GA NAI. ':= ' O OGINATTA.	:= をおきなう	switch A, B, C;
225	5	行先式の並びで区切り記号がま ちがっている	YUKISAKISIKI NO NARABI(SWITCH LIST) NO KUGIRIKIGO(DERIMITER) GA MATIGATE IRU. ', ' TO MINASITA.	, とみなす	switch A:= L ₁ , L ₂ *L ₃ ;

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
226	7	宣言文がまちがっている	SENGEN(DECLARATION) GA MATIGATTE IRU. KUUBUN(DUMMY STATEMENT) TO MINASITA.	その宣言文を無視する	real ;
227	5	文の終りが;又はendでない	BUN GA ';' MATAWA 'END' DE OWATTE INAI. ';' KA 'END' MADE YOMITOBASU.	;かendまで読飛ばす	go to L, go to M;
228	5	文の頭に不適当な区切り記号がある	BUN NO ATAMA NO KUGIRIKIGO(DELIMITER) GA HUTEKITO.YOMITOBASU.	読飛ばす	...; step A,B;
229	7	if then if	'THEN' NO TUGI NI JOKENBUN GA ARU. KUBUN TO MINASU.	空文とみなす	if ... then if...;
230	7	thenに続く文の次に;又はelseがない	'THEN' NO TUGI NO BUN NO ATO GA ';' KA 'ELSE' DENAI.MUSISURU.	読飛ばす	if then go to L1 go to L2;
231	7	繰返し文の制御変数の次に:=がない	KURIKAESIBUN NO SEIGYOHENSUU NO TUGI NI '=' GA NAI. TUGINO ';' KA 'END' MADE O KUUBUN TO MINASU.	;かendまで読飛ばす	for I, 1, 2 do;

エラー 番号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
232	7	繰返し文の繰返し要素が正しく 書けていない	KURIKAESI YOSO NI AYAMARI GA ARU. TUGINO ' ; ' KA 'END' MADE O KUUBUN TO MINASU.	; か end まで脱飛 ばし	for I := 1 step 1 do
233	9	コンパイラー・エラー	'COMPILER' NI 'ERROR' GA OKITA.	コンパイル打ち切り	
234	2	名前の二重定義	ONAJI NAMA E GA 1 TU NO 'BLOCK' DE 2 DO SENGEN SARETE IRU. 1 PO TO MUSI SURU.	一方を無視する	real A, A;
235	9	コンパイラー・エラー	'COMPILER' NI 'ERROR' GA OKITA.	コンパイル打ち切り	
236	9	コンパイラの working file へ の I/O におけるエラー	'MAGNETIC' DISK' ENO NYUSYUTU -- RYOKU NI 'ERROR' GA OKITA.	"	
237	10	"	209 と同じ	"	
238	7	条件節に if が無い	ZYOKENSETU (IF CLAUSE) NI 'IF' GA NAI.		
239	3	条件節に then が無い	ZYOKENSETU (IF CLAUSE) NI 'THEN' GA NAI. 'THEN' O OGINATTA.	then を補った	

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
240	4	条件節に論理式以外のものが現れた	ZYOKENSETU(IF CLAUSE) NI RON- RISIKI DE NAI SIKI GA KI TA. ZERO O 'TRUE' TO SITA,	零のとき true, 零でないとき false とした	
241	4	then に対する else がない	'THEN' NI TAIU SURU 'ELSE' GA NAI. 'ELSE' O OGINATTA.	else を補った	
242	2	比較作用素の使い方が違っている	RONRI SIKI NI MATIGAI GA ATTU- TA. 'AND' O OGINATTA.	∧ を挿入した	$A < B < C \rightarrow A < B \wedge B < C$
243	4	算術式がくるべきところに別のものがきた。	SANDYUTU SIKI(ARITHMETIC EX- PRESSION) GA KURU TOKORO NI BE- TU NO MONO GA KITA.		if $A < B$ then $C < D$ else $E := F + C$;
244	3	算術作用素, 比較作用素, 論理作用素, 区切り記号が2つ以上続いていた。	(ARITHMETIC, RELATIONAL, LOGI- CAL OPERATOR) MATAWA (DELIMI- TER) GA 2 TU_IDYO TUZUITE KITA. 1 TU TO MINASITA.	1つあるとみなした	$A(X, Y) \rightarrow A(X, Y)$
245	9	コンパイラの処理間違い。	(COMPILER) NO MATIGAI.		
246	5	(, [, vec の位置が不適当	('(', '[', 'VEC' NO ITI GA MATI- GATTE ITA.		

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
247	4	作用素が欠けている	SAYOUSO(OPERATER) GA NAI. '＊' O IRETA.	＊を挿入した	real A ; A ()
248	5	左辺に式がある	SAHEN(LEFT PART) NI SIKI GA ARU.		A+B:= A+B+8 ;
249	5	[] の誤用	'['_NO MATIGAI.		A+[]
250	6	代入式の中に先行式があった	DAINYU SIKI(ASSIGNMENT STATE- MENT) NO NAKA NI YUKISAKI SI- KI(DSIGNATIONAL EXPRESSION) GA KITA		
401	9	スタックがあふれた	'BLOCK' GA OOKI SUGIRU. 'STACK' GA AHURETA.	コンパイル打ち切り	
402	9	一つのブロックで使っている変 数が多すぎる	'BLOCK' DE TUKATTE IRU NAMAЕ (IDENTIFIER) GA OOSUGIRU.	//	
408	9	定数や記号例が多すぎて、コン スタント・テーブルがあふれた	TEISU(CONSTANT) YA KIGORETU (STRING) GA OOSUGIRU. 'CONSTANT TABLE' GA AHURETA.	//	

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
404	9	コンパイラーのエラー	'CONPILER ERROR' GA OKITA,	コンパイル打切り	
405	9	コンパイラのworking file へ の I/Oにおけるエラー	209と同じ	"	
406	9	コンパイラのworking file へ の I/Oにおけるエラー	209と同じ	"	
407	9	コンパイラ・エラー	208と同じ	"	
408	9	"	208と同じ	"	
409	9	"	208と同じ	"	
410	9	コンパイラのworking file へ の I/Oにおけるエラー	209と同じ	"	
411	9	コンパイラのworking file へ の I/Oにおけるエラー	209と同じ	"	
412	9	external name table があふれた (100個以上)	'EXTERNAL NAME TABLE' GA AHU- RETA.	"	

エラー 番 号	grade	原 因	エ ラ ー ・ メ ッ セ ー ジ	処 理	例
			1 10 20 30 40 50		
413	9	vector expression で左辺が array でない	BEKUTORU SIKI(VECTOR EXPRESSI- ON) DE SAHEN(LEFT PART) GA 'ARRAY' DE NAI.		real B ; B:=vec tor ;
414	9	右辺に vec がない	UHEN(RIGHT PART) NI 'VEC' GA NAI.		array A ; A[*,*]:=1 tor;
500	9	コンパイラの working file へ の I/O におけるエラー	209 と同じ	コンパイル打切り	
501	9	"	209 と同じ	"	

ALGOL (DISK) RUN TIME ERROR

G-600	実パラメタが仮パラメタの数より少ない	※
G-601	" 多い	※
G-602 ~607	名前で呼ばれるパラメタのタイプが一致してない	
G-608 ~609	値どりのパラメタのタイプが一致していない	
G-610	配列の宣言において、上限<下限となつた	
G-611	READ の実パラメタに不適当なものがある	
G-614 ~615	スタックのあふれ	
G-616	値の定義されていないスイッチを用いた	
G-620	double integer への変換でオーバーフローが起きた	
G-625	データとして正しくない文字を読んだ0として続行	※
G-626	数値データとして正しくないものを読んだ "	※
G-630	数値データの中に ' . ' が二つある	
G-631	" 指数部以外で ' - ' が二つある	
G-632	" 指数部に符号が二つある	
G-633	" 10(E) が二つある	
G-635	DATA 不足	
G-660	ownによりエリアがオーバーフローした	
G-670	PUTD, GETD にてハード的エラーが起きた	
G-671	PUTD をしていないN(レーベル)をGETDしている	
G-672	PUTD で DISK があふれた	
G-673	GETD でハード的エラーが起きた	

注1) ※印は RUN 続行