解　説

FORTRAN プログラマからみた Ada†

西野秀毅†† 田辺裕一††† 山崎　順†††
上原憲二†††† 土田耕祚†††††

1. はじめに

FORTRAN 77[1] を知っているかまたは使っているプログラマが Ada 言語[2] を理解するのに役立つように，簡単な例題を両方の言語でコーディングし，それを比較することによって，FORTRAN と比較したときの Ada の特徴を説明する．Ada 言語の基本的な構文や特徴的な機能のいくつかを FORTRAN と対応させて述べている．二つの例題を通してプログラムの書き方，全体構造，代入，制御文，手続き，関数などの基本的なことと，パッケージ，例外などの FORTRAN 言語にない Ada 言語の特徴的な項目について説明する．

2. 例 1（列挙型と例外処理の利用）

2.1 問　題

端末から自分と相手の星座を読み込んで，相性判断を画面に表示する．相手の星座は最高 12 まで入力できる．星座は 12 通りあり，よい相性，悪い相性はそれぞれ 120°，90° 離れた星座同士である．また，0° または 180° 離れた星座同士は両極端，他の場合は普通の相性である．たとえば，おひつじ座と相性のよい星座は，しし座，いて座である．

2.2 FORTRAN での記述

問題の FORTRAN 77 によるコーディングを図-1 に示す．

2.3 Ada での記述

問題の Ada によるコーディングを図-2 に示す．

† Ada for FORTRAN Programmers by Hideki NISHINO (Systems Development Laboratory, Hitachi, Ltd.), Yuichi TANABE, Jun YAMASAKI (Software Works, Hitachi, Ltd), Kenji UEHARA (Information Systems and Electronics, Mitsubishi Electric Corporation), Kousaku TSUCHIDA (Kamakura Works, Mitsubishi Electric Corporation).
†† (株)日立製作所システム開発研究所
††† (株)日立製作所ソフトウエア工場
†††† 三菱電機(株)情報電子研究所
††††† 三菱電機(株)鎌倉製作所

```
01  C
02  C      AISHO HANDAN
03  C
04         CHARACTER*8   ANATA, SAP (12, 2)
05         INTEGER       SEIZSU
06         PARAMETER     (SEIZSU=12)
07  C
08         OPEN (5, BLANK='NULL')
09       100 CONTINUE
10         WRITE (6, ' (1H1, 10X, " ** AISHO HANDAN **") ')
11         WRITE (6, ' (1H, " (ANATA NO SEIZA =>") ')
12         READ (5, '(A8)', END=1400) ANATA
13         WRITE (6, '(1H, "AITE NO KAZU (12 INAI) =>") ')
14         READ (5, '(I2)' ) N
15         IF (.NOT. ((N.GE. 1) .AND. (N.LE.SEIZSU))) GOTO 1600
16         WRITE (6, '(1H, " AITE NO SEIZA =>") ')
17         DO 200 I=1, N
18           READ (5, '(A8)') SAP (I, 1)
19       200 CONTINUE
20         CALL HANDAN (ANATA, SAP, N, *1500)
21         WRITE (6, '(1H, "                ** HANDAN KEKKA **") ')
22         WRITE (6, '(1H, " ANATA NO SEIZA: ", A8/
```

```
      &          1H, " AITE NO SEIZA TO AISHO: ") ') ANATA
        DO 300 I=1, N
          IF (SAP (I, 2) .EQ. 'YOI') THEN
            WRITE (6, '(1H, 10X, A8, " NAKAYOKU DOOZO.") ') SAP (I, 1)
          ELSE IF (SAP (I, 2) .EQ. 'WARUI') THEN
            WRITE (6, '(1H, 10X, A8, " WAKARETA HOOGA YOI.") ') SAP (I, 1)
          ELSE IF (SAP (I, 2) .EQ. 'KYOKUTAN') THEN
            WRITE (6, '(1H, 10X, A8, " MIKIWAME GA DAIJI.") ') SAP (I, 1)
          ELSE IF (SAP (I, 2) .EQ. 'FUTSU') THEN
            WRITE (6, '(1H, 10X, A8, " DORYOKU SHIDAI DESU.") ') SAP (I, 1)
          END IF
  300   CONTINUE
        GOTO 100
 1400   WRITE (6, '(1H, "          ** OWARI DESU **") ')
        STOP
 1500   WRITE (6, '(1H, " SEIZA MEI AYAMARI DESU.") ')
        GOTO 100
 1600   WRITE (6, '(1H, " AITE NO KAZU AYAMARI DESU.") ')
        GOTO 100
        END
C
C-------------------------------------------------------------------------------
C
        SUBROUTINE HANDAN (ANATA, SAP, N, *)
        CHARACTER*8  SEIZA (12), ANATA, SAP (12, 2), LA
        INTEGER      SEIZSU
        PARAMETER    (SEIZSU=12)
        DATA SEIZA   /'OHITSUJI', 'OUSHI'    , 'FUTAGO'
       &           , 'KANI'   , 'SHISHI'    , 'OTOME'
       &           , 'TENBIN' , 'SASORI'    , 'ITE'
       &           , 'YAGI'   , 'MIZUGAME', 'UO'/
        DO 2000 I=1, 12
          IF (SEIZA (I) .NE. ANATA) GOTO 2000
            KPS=I
            GOTO 2100
 2000   CONTINUE
        RETURN 1
 2100   DO 2400 I=1, N
          DO 2200 J=1, 12
            IF (SEIZA (J) .NE. SAP (I, 1)) GOTO 2200
              KPSA=J
              GOTO 2300
 2200     CONTINUE
          RETURN 1
 2300     IF ((KPSA .EQ. (MOD ((KPS+ 3), SEIZSU)+1)) .OR.
     &       (KPSA .EQ. (MOD ((KPS+ 7), SEIZSU)+1))) THEN
            LA='YOI'
          ELSE IF ((KPSA .EQ. (MOD ((KPS+ 2), SEIZSU)+1)) .OR.
     &             (KPSA .EQ. (MOD ((KPS+ 8), SEIZSU)+1))) THEN
            LA='WARUI'
          ELSE IF ((KPSA .EQ. (MOD ((KPS- 1), SEIZSU)+1)) .OR.
     &             (KPSA .EQ. (MOD ((KPS+ 5), SEIZSU)+1))) THEN
            LA='KYOKUTAN'
          ELSE
            LA='FUTSU'
          ENDIF
          SAP (I, 2)=LA
 2400   CONTINUE
        RETURN
        END
```

図-1 FORTRAN 77 言語によるコーディング

```
01 with TEXT_IO; use TEXT_IO;
02 procedure AISHO_HANDAN is
03   type SEIZA is (OHITSUJI, OUSHI      , FUTAGO   ,
04                  KANI    , SHISHI     , OTOME    ,
05                  TENBIN  , SASORI     , ITE      ,
06                  YAGI    , MIZUGAME, UO       );
07   type AISHO is (YOI, WARUI, KYOKUTAN, FUTSU);
08   SEIZA_SU: constant :=12;
09   type HANI is range..SEIZA_SU;
10   type SEIZA_ARRAY is array (HANI range <>) of SEIZA;
11   type AISHO_ARRAY is array (HANI range <>) of AISHO;
12   type SEIZA_AISHO_RECORD (SIZE: HANI) is
13     record
14       AITE: SEIZA_ARRAY (1..SIZE);
15       KEKKA: AISHO_ARRAY (1..SIZE);
16     end record;
17   type SEIZA_AISHO_P is access SEIZA_AISHO_RECORD;
18   package SEIZA_IO is new ENUMERATION_IO (ENUM => SEIZA);
19   use SEIZA_IO;
20   package INT_IO is new INTEGER_IO (NUM => INTEGER);
21   use INT_IO;
22   ANATA: SEIZA;
23   SAP: SEIZA_AISHO_P;
24   H: HANI;
25   N: INTEGER;
26   AITE_SU_ERROR: exception;
27
28   procedure HANDAN (PS: in SEIZA; PSA: in SEIZA_ ARRAY;
29                     PAA: out AISHO_ARRAY              ) is separate;
30
31 begin
32   SEIZA_IO. DEFAULT_SETTING := LOWER_CASE;
33   loop
34     NEW_PAGE;
35     PUT_LINE ("          ** AISHO HANDAN **");
36     begin
37       PUT_LINE (" ANATA NO SEIZA ==>"); GET (ANATA); NEW_LINE;
38       PUT_LINE (" AITE NO KAZU (12 INAI) ==>"); GET (N); NEW_LINE;
39       if 1 <= N and then N <= SEIZA_SU then
40         H := HANI (N);
41       else
42         raise AITE_SU_ERROR;
43       end if;
44       SAP := new SEIZA_AISHO_RECORD (H);
45       PUT_LINE (" AITE NO SEIZA ==>");
46       for I in 1..H loop
47         GET (SAP. AITE (I));
48       end loop;
49       HANDAN (ANATA, SAP. AITE, SAP. KEKKA);
50       NEW_LINE;
51       PUT_LINE ("          ** HANDAN KEKKA **");
52       PUT (" ANATA NO SEIZA: "); PUT (ANATA); NEW_LINE;
53       PUT_LINE (" AITE NO SEIZA TO AISHO: ");
54       for I in 1..H loop
55         SET_COL (10); PUT (SAP. AITE (I)); SET_COL (19);
56         case SAP. KEKKA (I) is
57           when YOI      => PUT_LINE ("NAKAYOKU DOOZO. ");
58           when WARUI    => PUT_LINE ("WAKARETA HOOGA YOI. ");
59           when KYOKUTAN => PUT_LINE ("MIKIWAME GA DAIJI. ");
60           when FUTSU    => PUT_LINE ("DORYOKU SHIDAI DESU. ");
61         end case;
```

```
62        end loop;
63      exception
64        when DATA_ERROR => PUT_LINE (" SEIZA MEI AYAMARI DESU. ");
65        when AITE_SU_ERROR => PUT_LINE (" AITE NO KAZU AYAMARI DESU. ");
66      end;
67    end loop;
68  exception
69    when END_ERROR => PUT_LINE ("          ** OWARI DESU **");
70  end AISHO_HANDAN;
71
72  ------------------------------------------------------------------------------
73
74  separate (AISHO_HANDAN)
75  procedure HANDAN (PS: in SEIZA; PSA: in SEIZA_ARRAY;
76                    PAA: out AISHO_ARRAY      ) is
77    KPS, KPSA: INTEGER range 0..SEIZA_SU-1;
78    LA: AISHO;
79  begin
80    KPS := SEIZA'POS (PS);
81    for I in PSA'RANGE loop
82      KPSA := SEIZA'POS (PSA (I));
83      if KPSA = (KPS+ 4) mod SEIZA_SU or else
84         KPSA = (KPS+ 8) mod SEIZA_SU then
85        LA := YOI;
86      elsif KPSA = (KPS+ 3) mod SEIZA_SU or else
87            KPSA = (KPS+ 9) mod SEIZA_SU then
88        LA := WARUI;
89      elsif KPSA = (KPS+ 0) mod SEIZA_SU or else
90            KPSA = (KPS+ 6) mod SEIZA_SU then
91        LA := KYOKUTAN;
92      else
93        LA := FUTSU;
94      end if;
95      PAA (I) := LA;
96    end loop;
97  end HANDAN;
```

図-2 Ada 言語によるコーディング

2.4 解　説

（1） 列挙型の入出力

SEIZA は値の集合が 12 個の星座からなる型を表す（行 3-6）．行 18 は SEIZA 型の入出力パッケージ SEIZA_IO を列挙型入出力汎用体パッケージの具体化によって実現している．入出力される文字列が SEIZA 型に適っているかは SEIZA_IO 中の入出力手続き（たとえば行 37 の GET）が検査する（参考文献 3) 参照）ので，その検査のための文を書く必要はない．

（2） 上下限が不定な配列

行 11 は SEIZA 型を要素とする配列型の定義であるがその上下限が 1 から 12 以内であれば良いことを表している．これは相性判断の相手数は診断ごとに異なることに対応している．行 12-16 は相手の星座と判断結果を表す配列を要素にもつレコード型であり，要素配列の長さは相手数を表す判別子により表している．これらにより長さが固定でない配列を扱っている．行 40 で相手数が決定された後，行 44 で配列長がちょうど相手数分である SEIZA_AISHO_RECORD 型の算体が作り出される．行 47 で相手の星座を入力して実際に相性判断する手続き HANDAN に渡す（行 49)．手続き HANDAN の宣言は行 28-29 にあるがその全体は行 74-97 に分離されている．HANDAN 内では各相手に対し判断するために RANGE 属性を用いている（行 81)．これは与えられた配列の下限から上限の範囲を表す．長さが固定でない配列を処理するために，FORTRAN では長さもパラメタで渡したが，Ada では不要となっている．しかも，ドキュメント性が良くなっている．

（3） 例外の利用

相性判断を繰返し行うために loop 文を用いている（行 33-67)．このループは無限ループの形をしており，

これから出る，すなわち相性判断を終えるために例外 END-ERROR を利用している．星座や相手数の入力の際にファイル終端子が認識されると例外 END-ERROR が発生し，制御が例外ハンドラ（行 69）に移ることによってループから出る．

入力誤りを各診断ごとに処理するためにブロック文を用いている．星座の入力において（行 3-6）の宣言にないつづりが入力されるとその検査において誤りとなり例外 DATA-ERROR が発生され，制御はそのブロック文の対応する例外ハンドラ（行 64）に移る．すると星座名誤りのメッセージが出力され，制御はブロック文を出る．また，相手数の入力誤りに対しては行 39 の if 文による検査で誤りとなったときユーザ定義（行 26）の例外 AITE-SU-ERROR を発生させ（行 42）て対処している．これも整数型汎用体パッケージを HANI 型に対して具体化すれば，入力手続きで検査するようにできるが，その場合発生するのは例外 DATA-ERROR なので，星座名誤りなのか相手数誤りなのか区別できない．

このように例外を用いると誤りに対する処理が本来の処理から分けられるのでプログラムがすっきりし見やすくなる．

（4） 副単位の利用

行 28-29 の宣言は手続きの仕様だけを示し，その本

```
      INTEGER BUNSU1(2), BUNSU2(2), X(2), Y(2)
      EXTERNAL PLUS, MINUS
      BUNSU1(1)=1
      BUNSU1(2)=2
      BUNSU2(1)=1
      BUNSU2(2)=3
      CALL MINUS (BUNSU1, BUNSU2, X)
      PRINT 100, 'X=', X(1), X(2)
      BUNSU1(1)=2
      BUNSU1(2)=4
      BUNSU2(1)=1
      BUNSU2(2)=5
      CALL PLUS (BUNSU1, BUNSU2, Y)
      PRINT 100, 'Y=', Y(1), Y(2)
  100 FORMAT (1X, I5, '/', I5)
      END

      SUBROUTINE PLUS (A, B, C)
      INTEGER A(2), B(2), C(2), D(2), E(2)
      EXTERNAL TSUBUN, YAKUBUN
      D(1)=A(1)
      D(2)=A(2)
      E(1)=B(1)
      E(2)=B(2)
      CALL TSUBUN (D, E)
      C(1)=D(1)+E(1)
      C(2)=D(2)
      CALL YAKUBUN (C)
      END

      SUBROUTINE MINUS (A, B, C)
      INTEGER A(2), B(2), C(2), D(2), E(2)
      EXTERNAL TSUBUN, YAKUBUN
      D(1)=A(1)
      D(2)=A(2)
      E(1)=B(1)
      E(2)=B(2)
      CALL TSUBUN (D, E)
      C(1)=D(1)-E(1)
      C(2)=D(2)
      CALL YAKUBUN (C)
      END

      SUBROUTINE GCD (I, J, K)
      L=I
      M=J
    1 IF (L.EQ. 0) THEN
        K=M
        RETURN
      ELSE IF (M.EQ. 0) THEN
        K=L
        RETURN
      ELSE IF (L.GE. M) THEN
        L=MOD (L, M)
      ELSE
        M=MOD (M, L)
      END IF
      GOTO 1
      END

      SUBROUTINE LCM (I, J, K)
      EXTERNAL GCD
      CALL GCD (I, J, L)
      K=I/L*J
      END

      SUBROUTINE TSUBUN (A, B)
      INTEGER A(2), B(2)
      EXTERNAL LCM
      CALL LCM (A(2), B(2), L)
      A(1)=A(1)*(L/A(2))
      A(2)=L
      B(1)=B(1)*(L/B(2))
      B(2)=L
      END

      SUBROUTINE YAKUBUN (A)
      INTEGER A(2)
      EXTERNAL GCD
      CALL GCD (A(1), A(2), K)
      A(1)=A(1)/K
      A(2)=A(2)/K
      END
```

図-3 FORTRAN 言語による分数プログラム

体を別の翻訳単位にすることを示している．ここでは副単位の例を示すために用いたが，むやみに用いると翻訳単位が多くなり管理が複雑になるなど弊害もあるので注意が必要である．有効的な利用法の一つとして，内側にネストする副プログラムが多い場合，外側の宣言部と実行部が大きく隔たってしまい読みにくくなることを防ぐために用いることがある．

3. 例 2（パッケージの利用）

3.1 問　題

分数の通分，約分，加算，減算用の演算ルーチンを作成する．この演算ルーチンを作成するに当って最大公約数(GCD)，最小公倍数(LCM)を求める演算ルーチンも必要となるのでそのルーチンも同時に作成する．ここでは帯分数は扱わない．

3.2 FORTRAN での記述

問題の FORTRAN 77 コーディングを図-3 に示す．

3.3 Ada での記述

問題の Ada によるコーディングを図-4 に示す．

3.4 解　説

（1） プログラム構造と分離翻訳

プログラムの仕様とその実現方法を明確に区別することはモジュラプログラミングの基本概念であり Ada ではこの概念に沿ったプログラム開発ができる．本プログラム例では分数の定義をパッケージ仕様（1行目―13行目）で表現し，分数の実現はパッケージ本体（15行目―70行目）で表現した．分数型を参照する主プログラムは副プログラム（72行目―78行目）として表現した．これら3つのプログラム単位はそれぞれ別々に翻訳できる．さらに分離翻訳機能としてこれらプログラム単位間にわたる識別子の定義，参照に対して，翻訳時にこれらの単位を調べ文法上の意味検査やオブジェクト生成などを行う．

（2） パッケージ仕様を用いた分数型の定義

Ada ではプログラムの汎用性を高めるための抽象データ型を定義することができる．

```
01 package BUNSU-ENZAN is
02   type BUNSU is private;
03   function "/" (P, Q: INTEGER) return BUNSU;
04   function "+" (I, J: BUNSU) return BUNSU;
05   function "-" (I, J: BUNSU) return BUNSU;
06   procedure PRINT (I: BUNSU);
07 private
08   type BUNSU is
09     record
10       BUNSHI : INTEGER;
11       BUNBO  : INTEGER;
12     end record;
13 end BUNSU-ENZAN;
14
15 with TEXT-IO; use TEXT-IO;
16 package body BUNSU-ENZAN is
17   package IO-INTEGER is new INTEGER-IO (INTEGER);
18   use     IO-INTEGER;
19   function GCD (P, Q: NATURAL) return POSITIVE is
20   begin
21     if P=0 then return Q;
22     elsif Q=0 then return P;
23     elsif P)=Q then return GCD (P rem Q, Q);
24     else return GCD (P, Q rem P);
25     end if;
26   end GCD;
27   function LCM (P, Q: POSITIVE) return POSITIVE is
28     begin
29       return P/GCD (P, Q)*Q;
30     end LCM;
31   procedure TSUBUN (I, J: in out BUNSU) is
32     T: POSITIVE;
33   begin
34     T:=LCM (I. BUNBO, J. BUNBO);
35     I:=(I. BUNSHI*(T/I. BUNBO), T);
36     J:=(J. BUNSHI*(T/J. BUNBO), T);
37   end TSUBUN;
38   function YAKUBUN (I: in BUNSU) return BUNSU is
39     R: POSITIVE;
40   begin
41     R:=GCD (abs (I. BUNSHI), I. BUNBO);
42     return (BUNSHI=> I. BUNSHI/R, BUNBO=> I. BUNBO/R);
43   end YAKUBUN;
44   function "/" (P, Q: INTEGER) return BUNSU is
45   begin
46     if Q)0 then return (BUNSHI=>P, BUNBO=>Q);
47     else return (BUNSHI=>-P, BUNBO=>-Q);
48     end if;
49   end "/";
50   function "+" (I, J: BUNSU) return BUNSU is
51     K, L: BUNSU;
52   begin
53     K:=I; L:=J;
54     TSUBUN (K, L);
55     return YAKUBUN (BUNSHI=>K. BUNSHI+L. BUNSHI,
56     BUNBO=>K. BUNBO);
57   end "+";
58   function "-" (I, J: BUNSU) return BUNSU is
59     K, L: BUNSU;
60   begin
61     K:=I; L:=J;
```

```
62      TSUBUN (K, L);
63      return YAKUBUN (BUNSHI=>K.BUNSHI-L.BUNSHI,
64      BUNBO=>K.BUNBO);
65    end "-";
66    procedure PRINT (I: BUNSU) is
67    begin
68      NEW_LINE; PUT (I.BUNSHI); PUT ("/"); PUT (I.BUNBO);
69    end PRINT;
70  end BUNSU_ENZAN;
71
72  with BUNSU_ENZAN; use BUNSU_ENZAN;
73  procedure MAIN is
74  begin
75    X, Y: BUNSU;
76    X:=1/2-1/3; PRINT (X);
77    Y:=2/4+1/5; PRINT (Y);
78  end MAIN;
```

図-4 Ada 言語による分数プログラム

ここではその機能を使って分数型を作成する方法を示す.

分数型の名前を BUNSU とし，密閉型として宣言（2行目）する．ここでの分数は帯分数でないため2つの整数部分から構成される．このため図-3の FORTRAN プログラムでは分数を配列として宣言した．Ada プログラムにおいては分数型のデータ構造として記述性を重視しレコード型とした（8行目—12 行目）．パッケージ仕様部には可視部と密閉部があり，このパッケージ仕様部と本体部以外の翻訳単位では可視部で宣言された部分のみが参照でき，密閉部の宣言項目は参照できない．分数型に対して，2つの整数から分数を作りだす演算子，分数の加減算の演算子を定義した（3行目—5行目）.

（3） パッケージ本体を用いた分数型の実現

抽象データ型の実現内容は，この型を使うユーザプログラムからは参照できないパッケージ本体で記述する．したがってこの本体の変更が生じても，ユーザプログラムの変更の必要はない.

（4） 機能及び記述性の良さ

副プログラム(手続き及び関数)の再帰呼出しができる（24 行目）．複合型算体に対する記述性が優れている（35, 36, 42行目など）．さらに関数のリターン値として複合型算体が許されている（42 行目）．演算子記号を文法が定義している意味とは別の意味に定義することができる．さらに関数名や演算子の多義が許される．35行目と 76行目の "/" は異なったものである.

4. Ada 使用にあたって

言語を有効に使用するにあたって重要な点は，利用者が言語の狙い，特徴をプログラム開発時に生かすことである．特に留意すべき点を以下に示す.

（1） 汎用性の高いソフトウェア部品の作成，活用が既存の言語よりも容易である．このためには機種非依存なデータ型，内部構造に非依存な抽象データ型，名前を抽象化した“名前の多義定義”，プログラムの仕様と実現の分離，パッケージや副プログラムを抽象化した汎用体などを活用するとよい.

（2） 信頼性の高いプログラム作りが既存言語より容易である．Ada では翻訳時やプログラムの実行時に高度なエラー検出ができるため，部分型，派生型，例外ハンドラなどの使用によりこれらを活用するコーディング方法を取るとよい.

（3） プログラムのモジュール分割や副単位の活用によってトップダウン開発方式の採用がより容易である.

（4） システムプログラム記述においては，非同期処理，絶対番地参照，ビット処理，特権命令の使用，一部高性能化のための機械語の使用などのためアセンブラの使用が不可避であったが Ada においては，高水準のタスク機能，内部表現節などによってほとんどの部分を Ada で記述できるようになった.

5. おわりに

本稿では，FORTRAN 77 プログラムと Ada プログラムを対比させて，Ada プログラムの記述方法を説明した.

FORTRAN は，科学技術計算用としてもっとも普及しているプログラミング言語であり，科学技術計算処理向きの機能と長年の積み重ねによる膨大な資産を有している．Ada には，まだそのような資産はないが，Ada言語の特徴であるパッケージを整備することによって，FORTRAN にとってかわるものと思われる．特にその読みやすさ，エラーチェックの強力さ，豊富な機能は，開発・保守を容易にするであろう．今後，大規模な，ライフサイクルの長いソフトウェアに Ada を適用することにより，ソフトウェアのライフサイクル・コストを下げることができるものと期待している.

謝辞 本稿をまとめるにあたっては Ada 分科会の主査である NTT の福山峻一調査役をはじめ

Ada 分科会のメンバ各氏のご指導，ご支援を得たので，ここに感謝する．

参考文献

1) Programming Language FORTRAN, ANSI X 3.9 (1978).
2) U. S. Department of Defence : Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A (1983)；邦訳-情報処理振興事業協会(編)：最新 Ada 基準文法書，bit 別冊，共立出版 (1984).

(昭和 60 年 12 月 9 日受付)