解　説

FORTRAN 77 の特徴†

——JIS FORTRAN との非互換性の観点から——

西　村　恕　彦††

1. FORTRAN 66 から FORTRAN 77へ

筆者はかつて FORTRAN IV 言語を学会誌上に解説したことがあった[1]が，それから 17 年を経て，今度は FORTRAN 77 を紹介することになった．

FORTRAN はもともと，IBM 704 用の算譜言語であったのが，しだいに広く作成・利用されるようになり，アメリカ規格協会がそれらの言語仕様の共通部分を定める形で第一次の規格化を行った．制定されたのは 1966 年のことであり，その言語は現在，FORTRAN 66 と呼ばれている．

これは算譜言語の規格化として最初のものであったので，さまざまの困難が伴ったが，また，以後の他の言語の規格化にも大きな影響を与えた．

規格化にあたって，FORTRAN は過渡的な言語であり，ある期間の後は死語になると想定された．したがって，FORTRAN 66 は閉じた言語として設計されたし，アメリカ規格協会が行った二回の保守[2,3]は，明確化という消極的なものであった．

しかし FORTRAN は死語にならず，アメリカ規格協会は拡張への針路をとった．FORTRAN 77 はもはや既存の言語仕様の共通集合ではない．それは全体として「良い言語」を目指しており，新しく設計された機能も多い．

第 2 階の定義 (JIS 10.2.9) や関数引用の最適化 (JIS 10.2.10) のようなわかりにくい概念は削除され，より単純で自明な言語となった．

また，FORTRAN は今後とも生き続ける言語と考えられ，その固有の発展の一つの区切りとして設計された．将来の拡張を念頭におき，その障害になる仕様は削除する方針がとられた．その結果，旧規格からの上向きの互換性は完全には保証されていない．

FORTRAN 77 の全体像[4,5,6]はまだあまり紹介されていないが，それをここで試みることは適当ではあるまい．以下では，JIS FORTRAN 水準 7000 との非互換性という視点から論じておこう．

なお，“77” という名称は，単なる番号であるとされているが，1977 年制定の見込みであらかじめ付けたものであることに，間違いはない．実際の制定は，翌 1978 年 4 月 3 日である．

規格は二水準からなる．上位は単に FORTRAN または full language と呼ばれ，水準 7000 の拡張である．下位は subset FORTRAN と呼ばれ，水準 5000 相当である．両者は 1 冊の規格票の見開き，右ページ（上位）と左ページ（下位）に対照して記述されている．

2. FORTRAN 77 の技術的特徴

ここでは，FORTRAN 77 の文法に合った正しい算譜を，JIS FORTRAN 水準 7000 の文法に合った正しい処理系に掛けて，それが文法通り正しく動作しない場合に，どのように書き換えることができるか，について，主要な事項を述べる．

また，JIS FORTRAN 水準 7000 に限定してしまうことは，必ずしも日本の現状に合わないので，いわゆる大形機に既存の処理系言語のことも，漠然と対照することにする．

これらによって，FORTRAN 77 に特有の拡張部分が明示されることになろう．

2.1 DO 繰返しはゼロ回から始まる

DO 形並びおよび DO 文における繰返し回数の最低は，ゼロ回である．ゼロ回であるような文の実行は，JIS 規格では不定とされているので，その書換えをしなければならない．DO 文の書換えはよく知られているように簡単であるが，DO 形並びの書換えは煩雑である．例えば次のようなグラフの出力を考えてみよ．

```
      WRITE (6,6)('−', I=1,M), ('＊', J=1,N)
    6 FORMAT (1X, 120A1)
```

DO の母数は，整数，実数，倍精度数の算術式であり，負の増分も許される．算術式の値とそれから導かれる繰返し回数は，繰返しの開始時に一回だけ評価さ

† Incompatibilities with FORTRAN 77 by Hirohiko NISIMURA (Tokyo University of Agriculture and Technology).

†† 東京農工大学

れ，固定される．したがって，これを JIS 風に書き換えることは，容易である．

DO が満足されると，制御変数は“次の値”になっている．この点も，DO 文においては JIS 風に書き換えることが容易である．なお，FORTRAN 77 には第2階の不定の概念がないので，FORTRAN 77 では確定になっている値が，JIS では不定になっていることがありうる．

FORTRAN 77 の DO 繰返しの仕様は，整構造の算譜を構成するのにきわめて有効であることが体験されている．JIS の DO 繰返しは，そうではないのである．なお配列については，後出の **2.7** をもみよ．

2.2 段落 IF 文

段落 (block) IF 文については，次のような構文を掲げれば十分であろう．

```
IF (論理式) THEN
      [段落]
[ELSEIF (論理式) THEN
      [段落] ]…
[ELSE
      [段落] ]
ENDIF
```

これから論理 IF 文への書換えは，まったく単純な規則ですみ，文数も増えるわけではない．

文数が変わらないのならば逆に，段落 IF 文が導入されたことに，どのような意義があるだろうか．それは，余計な文番号を書かないことによって，制御構造が明示される点にあると考えられる．

2.3 副譜の入口

副譜 (subprogram) の入口は，SUBROUTINE 文，FUNCTION 文，ENTRY 文である．ENTRY 文の形は，次のとおりである．

```
ENTRY 入口名 [([仮引数並び])]
```

ENTRY 文は，JIS にはないが，多くの処理系にすでに実装されているので，変換上に大きな問題はないであろう．一つの副譜中に二つ以上の入口があるときに，それらのあいだで引数がどのように連合 (association) するか，ある仮引数がどの範囲で有効であるか，関数副譜名と入口名は連合するのかなど，仕様上のばらつきのあることが知られている．

FORTRAN 77 は，これらを制限の強い形で規定した．すなわち，仮引数を含んだ実行文を実行できるのは，その仮引数を含んだ SUBROUTINE 文，FUNCTION 文，ENTRY 文が引用されたときだけであり，さらに，副譜の実行文中で仮引数を書くことができるのは，それより上にある SUBROUTINE 文，FUNCTION 文，ENTRY 文のなかにその仮引数があるときだけである．また，関数副譜中のすべての入口名とその関数副譜名は，たがいに連合する（型が同じである必要はない）．

したがって，FORTRAN 77 の制限に従った正しい算譜であれば，あまり問題なく既存の処理系で処理できるであろう．しかし，これらの制限に従っていない算譜は，処理系間の移行がうまくゆかないこともありうる．

ENTRY 文のある算譜を，ENTRY 文のない処理系用に書き換えるのは，きわめて困難であろう．それは，単譜内ではどの入口点からも共用でき，しかも単譜に対して局所的な料 (data) の存在があるからである．手続きの共用だけが目的ならば，複製を用意すればすむのであるが，局所的な料の値を保存するような算法が採用されていると，JIS 規格では，局所性・保存性・共用性を三者とも満足するような機能がなく，対応できないのである．

2.4 局所料の保存

一般に副譜中の料 (data) は，有効範囲がその副譜に対して局所的であり，外部から参照できないというばかりでなく，制御がその副譜から戻ったときに，料の値がすべて不定になってしまう．

このことは，算譜の整構造化を困難にするとともに，情報の閉じ込め (information hiding) に対しても悪い影響を及ぼす．

副譜中に SAVE 文の宣言をおくことによって，局所的な料の値の保存を指示することができる．

この機能は JIS FORTRAN に書き換えることが不可能であるが，既存の処理系における記憶領域管理はどうなっているのであろうか．最近の操作系においては，副譜の実行が終わっても，局所的な料記憶の領域をそっくり保存するのが普通ではないか．もしそうであれば，算譜の書換えにあたって，SAVE 文の存在は無視してかまわないことになる．おそらく，現在の処理系作成者へのインパクトは，それほど大きくないであろう．

2.5 計算形戻り

計算形戻り (alternate return) に関係のある文は，例えば次のような形をしている．

```
CALL SUB (Y, *100, *200)
SUBROUTINE SUB (Z, *, *)
```

RETURN 整数式

算程 (subroutine) 中でこの RETURN 文を実行すると，整数式の値が評価され，その整数番目の星印に対応する文番号のところへ戻る．整数式の値が，星印の個数の範囲内にない場合（例えば負の場合）には，通常の戻りを行って，CALL 文の実行を完了する．

計算形戻りのある算譜を，それのない形に書き換えることは，簡単である．計算形戻りのある既存の処理系言語では，星印以外のアンド記号 (&) や ドル記号 ($) も使われてきたことを注意しておこう．

なお，計算形 GOTO 文の文法も，上記と同じような方式になっていて，整数式の値が，行き先の文番号の個数の範囲内にない場合には，次の文に進むことになる．この書換えも容易である．書換えを行って気持が悪いのは，すぐ上の文からしか制御が移ってこない，冗長な文番号を作らなければならないことである．

この点でも FORTRAN 77 の文法は，より自然なものであるといえる．

2.6 包括関数

基本外部関数の概念は廃止され，組込み関数に統合された．組込み関数名の多くは，手続き引用の引数として書くことができる．これが JIS の基本外部関数であれば問題はないが，JIS の組込み関数である場合には，書換えは煩雑な調整を伴う．

組込み関数の多くは，包括関数名 (generic name, 例えば LOG) と個別関数名 (specific name, 例えば ALOG, DLOG, CLOG) を有している．包括関数名から JIS の基本外部関数への書換え自体は容易である．（なおアメリカ規格協会では，将来は個別関数名が使われなくなり，包括関数名に統合されると予想している．）

組込み関数 INT は，値ゼロの方向に切捨てを行い，組込み関数 MOD も，このやりかたで剰余を求める．これらから最大の整数や一定符号の剰余を導くことは困難であるが，逆の誘導は容易である．上記の二つの関数は，旧規格からの悪しき遺産がそのまま継承された例である．

ただ FORTRAN 77 では，四捨五入を行う包括関数 NINT (nearest integer) があるので，これから導くことは可能である．

2.7 配 列

配列宣言子は，次の形である．

配列名 ([始点 :] 終点 [, [始点 :] 終点]…)

始点，終点は整数式であり，負の値も許される．配列は 7 次元まで許される．配列要素を参照するときの添数式は，整数式であって，関数引用や配列要素名を含んでいてもよい．ただし，副作用は禁止される．

なお，定数名を宣言する PARAMETER 文の形は，次のとおり括弧がある点で，いくつかの処理系言語の構文とは異なっている．

PARAMETER (定数名 = 定数式 [, 定数名 = 定数式]…)

これは，将来，英字名としてより長いつづりを許すようにするときのための調整である．

2.8 入出力について

入出力関係の拡張はきわめて広範囲なので，JIS 規格の処理系で対応することは，ほとんど不可能だろうと思われる．ここでは，ほんの二三の話題を拾うにとどめる．

書式仕様の走査中に **BN 編集記述子** に出会うと，以後の数値入力欄中の空白は，(ゼロではなく) 空とみなされる．これは，数値入力のデバッグ時によく知られているトラブルを回避するのに役立つと考えられる．ただしそのためならば，むしろ並び形書式 (list-directed formatting) を利用するほうが簡単だろう．

これは例えば次のように書く．

```
READ (5, *, END=800) A,B,C
```

入力料は，空白またはコンマで区切って並べた定数の形をしている．並びの終りを斜線で明示することもできる．コンマまたは斜線を使ったときには，定数を省略でき，それに対応する入力項目は代入されず，もとの値を保持する．(FORTRAN 77 には，NAME LIST 入出力はない．)

書式仕様の走査中にコロンに出会い，入出力並びがすでに尽きていると，走査がそこで終了する．これも JIS の欠点の一つをカバーする．例えば次のように書く．

```
      READ (5,5) K, (X(I), I=1,K)
    5 FORMAT (I10, 5F10.0 :/(10X, 5F10.0))
```

JIS では，コロンが書けないので省略したとすると，Kの値が 1〜4 のとき 1 枚，6〜10 のとき 2 枚，11〜15 のとき 3 枚のカードを読むことになる．ところが，Kの値がちょうど 5 のときに限って，1 枚目のカードが読まれ，2 枚目のカードがスキップされる．WRITE 文でも同様で，Kの値が 5 のときに限って，空白の行が一つできる．読み書きされない記録のスキップが，例外的に発生してしまう．

それが FORTRAN 77 でこのようにコロンを書くと，Kの値が 5 のときに，1 枚だけカードが読まれ，

スキップは起こらないことになるのである.

READ/WRITE 文では, 整数式の値によって外部装置の番号を指定するが, 整数式のかわりに文字型の変数名や配列名などを書くと, その文字列を書式仕様に従って「読み書き」することができる. これは, 多くの処理系では ENCODE/DECODE 文に書き換えることができる.

なお, FORTRAN 77 の入出力帖 (file) は, 次のように分類できる.

順呼出し ——内部帖 (上述の文字料)

直接呼出し——外部帖

2.9 文字型について

ホレリス型が削除され, 文字型 (character type) が導入された. 文字型の宣言は, 次のような形になる.

CHARACTER [*文字長 [,]] 並び

文字長は次のとおりである.

{整定数 / (整定数式) / (*)}

並びの要素は次のとおりである.

{英字名 / 配列宣言子} [* 文字長]

一部の処理系言語では, 文字型の配列の宣言は, 次のようなより「合理的」な形をとっていることを注意しておこう.

配列名*文字長 (次元指定)

文字式は次のとおりであって, 連結を表す.

文字式//文字要素

部分文字列 (substring) は次のとおりである. これは, 引用・代入・入出力などの対象にできる.

変数名 ([始点]:[終点])

配列名 (添数指定) ([始点]:[終点])

文字型の処理は以上でほとんど全部であるが, いずれも JIS 規格の処理系向けに書き換えることが困難である. しかし, 互換性のない副譜を個別的・網羅的に作ってゆく気になれば, とにかく処理だけはできよう.

文字料の代入や比較は, 原則として左端の文字位置をそろえ, 短いほうの文字列の右側に空白を補って行われる. ただ, 入出力欄の幅が入出力項目の文字長よりも広いときに限り, 右端の文字位置でそろえられる. これは, 初期の処理系と旧規格のなかにあった不統一性が, そのまま保存された例の一つである.

文字型の代入文について, 一つ, 注意すべき禁止規定がある. それは,「等号の左辺の代入される文字位置はすべて, 右辺の式で引用してはならない」というものである.

この種の禁止規定の議論は, 筆者が古くから取り上げている[7,8] 以外にはあまり見掛けないが, 代入の行われる対象が二つ以上の要素からなる集合である場合には, いつでも考慮しなければならないのである. いくつかの算譜言語が, なんらかの意味で集合を操作対象として含むようになりつつある動向からみて, この問題はもっと意識されてよいと思う.

上記の規定は, 次のような代入文を禁止することになるが, この禁止は十分に自明なことであるとは感じられない.

```
CHARACTER*1 A
CHARACTER*1000 STACK
    STACK=A//STACK
```

ここでは, 文字列の棚 STACK の左端に 1 文字 A を押し込み, 棚全体を右に 1 文字シフトしようとする. しかしこれでは, 左辺の代入される STACK が右辺の式で引用されることになるから, 禁止に該当するのである.

ちなみに, 棚からのポップアップは, 次のようにでも書けようが, これも文法違反である. (組込み関数 LEN は, 文字列の長さを与える.)

```
B=STACK
STACK=STACK(LEN(B)+1:)
```

3. JIS FORTRAN との矛盾

JIS FORTRAN 水準 7000 の文法に合った正しい算譜を, FORTRAN 77 の文法に合った正しい処理系に掛けて, それが文法通りに正しく動作しない場合に, どのように書き換えることができるか, について, 主要な事項を述べる. これについては, アメリカ規格の付録に, 一括して掲げられている.

このような縮小方向の改訂をなすべきではないという主張は, 国際的に繰り返しなされてきたが, 先述したように, FORTRAN の将来の発展を期待するならば, いまそれをなしておかなければならないのである. もちろん, 現在その項目はそれほど多いわけではない.

3.1 DO の拡張範囲

DO の拡張範囲は廃止された. DO の範囲からいったん外に出ると, 戻ってくることは禁止される. もちろん関数引用や CALL 文による移行は許される.

拡張範囲をもった算譜を一般的に FORTRAN 77

向きに書き換えることは，困難である．

3.2 ホレリス型

ホレリス型は廃止され，文字型が導入された．(ただし，H編集記述子だけは残された．)

ホレリス型から文字型への書換えは，それほど容易なものではない．JIS におけるホレリス型の扱いは，擬装 (guise) の概念のうえに構成されていて，それ自体，具合の悪い点が多かったのであるが，新規格では，文字型の確立に伴って，引用，代入，連合などは文字型同士に制限されている．したがって，擬装を利用して書かれた算譜は，そのままでは，絶対に FORTRAN 77 の処理系には掛からないし，機械的な構文の書換えによって変換することもできない．

もう一つ，FORTRAN 77 には，語胞 (numeric storage unit) と字胞 (character storage unit) を同じ共通区 (common block) のなかに混ぜて指定してはならないという制限がある．これは本質的な問題があるわけではないが，処理系間の移行性を保証するためのものである．これもその制限に従うように書き換えようとすると，大変である．

3.3 組込み関数

JIS では基本外部関数とされていたものが，組込み関数に統合されている．手続き引用の引数としてこれらの関数名を用いるときには，EXTERNAL 文による指定を，INTRINSIC 文に書き換えなければならない．

また，組込み関数名が何個か追加されたので，これらの名前と一致する作譜者定義の副譜名は，別のつづりに書き換えたり，EXTERNAL 文で指定したりしなければならない．

3.4 その他の縮小機能

JIS 規格では，初期の処理系の本質的ではない特性に由来するいくつかの奇妙な許容規定があった．これらは，算譜をより自然に書き，処理系の作成をより容易にするために，削除された．それには，次のようなものがある．

(1) JIS の EQUIVALENCE 文では，多次元の配列の要素を，配列要素関数に従う1次元の添字で参照することができた．

(2) 入出力並び中に無意味な括弧をそう入できた．

(3) 書式仕様中のH編集記述子を，文字列の入力に用いることができた．

3.5 副作用について

副作用という言葉を簡単に説明すると，ある文を実行したときに，陽に代入の記述されていない料の値が変更されてしまうことである．例えば次のような算譜の実行の効果は，副作用の典型的なものである[7]．

```
INTEGER FUNCTION K(N)
    K=N
    N=N+1
END
PROGRAM P
    REAL X(100)
    N=5
    X(K(N))=K(N)+10*K(N)
END
```

この主譜は，配列Xの一要素に対する代入を記述しているが，それを実行すると，主譜においては代入の記述されていない料Nの値が変更されてしまう．

FORTRAN 77 では，値を引用できるところでは広く式を書くことが許されるので，作譜者は，不注意に式を書いて，禁止されている副作用を発生しないように気を付けなければならない．

FORTRAN 77 には，「式の評価は，その式を含む文中の他の要素の値を変更するものであってはならない」という，きわめて包括的な規定がある．JIS 規格では，10.2.10 と 6.4 にあいまいな規定があっただけである．

筆者は，JIS があいまいに規定しているところは，すべて禁止的に解釈するのがよいと信じているが，もちろんこれとは別の立場もありうる．JIS 規格であいまいに規定されている事項を利用した算譜は，そのまま FORTRAN 77 で同じ結果を与えるとは限らない．

4. FORTRAN 82

アメリカ規格協会は，FORTRAN 77 を今後連続的に発展しつづけてゆく言語であるとみなし，次期に制定されるべき言語に対して，FORTRAN 82 という仮称を与えている．

それがどのような内容をもつことになるかは，数年たってみなければわからないわけであるが，現在，予想または検討されている事項には，次のようなものがある．

(1) 仕様を，機能単位・水準に分けて編成する．

(2) 整構造化のための機能を大幅に導入する．

(3) 旧規格にあった“悪い仕様”は，別枠にまとめる．

（4） データシステムズ言語協議会 (CODASYL) のデータベース機能[9] を導入する.

（5） パーデューの実時間言語（組込み算程）を導入する.

（6） その他.

参 考 文 献

1) 西村：FORTRAN IV 言語の概略，情報処理，1962-5.
2) USASI X3: Clarification of Fortran Standards —Initial Progress, *CACM*, 1969-5.
3) ANSI X3J3: Clarification of Fortran Standards —Second Report, *CACM*, 1971-10.
4) ANSI: Programming Language FORTRAN, X3.9-1978.
5) 西村：アメリカ規格 FORTRAN の改訂，**bit**, 1977-6.
6) 西村：人文科学の FORTRAN 77，東大出版会，1978-11.
7) 西村：JIS FORTRAN 全釈，オーム社，1974-5.
8) 西村：JIS COBOL 全釈，オーム社，1972-12.
9) 西村訳：FORTRAN データベース，共立出版，1978-10.
10) KWIC of FORTRAN 77 Full Language, ANSI X3J3, 1978.

（昭和54年3月15日受付）