To: J3 J3/24-139r2
From: John Reid & Hidetoshi Iwashita & Malcolm Cohen
Subject: 総称副プログラムの文法
Date: 2024-June-27
References: N2217, 23-223r2, 23-244r1.

1. Introduction

2023 年 6 月 12 日～16 日の会議において、WG5 は Fortran 202Y の N2217 に記述された総称副プログラム（generic subprogram）を承認することを決定した。この use cases は N2217 に記載されている。formal requirements は 23-233r2 で J3 によって承認された（24-147 で改訂）。formal specification は 23-244r1 で J3 によって承認された（24-48 で改訂）。ここでは syntax を示す。

2. Syntax

訳注: ここでは次の訳語を使用する。JIS 規格でこの訳がそのまま使われるとは限らない。

英語	日本語
generic subprogram	総称副プログラム
generic type declaration statement <generic-type-decl-stmt>	汎用型宣言文 汎用型宣言文
generic dummy argument type-or-kind generic <generic-type-spec> rank generic <generic-rank-spec> generic-dependent	汎用仮引数 型または種別について汎用的 汎用宣言型指定子 次元数について汎用的 汎用次元数指定子 汎用依存
kind generic <kind-generic-type-spec> <generic-*-spec>	種別について汎用的 種別汎用型指定子 汎用＊指定子
type generic <generic-type-specifier>	型について汎用的 汎用型指定子
<enumeration-type-spec>	enumeration 型指定子
<generic-rank> <generic-rank-range>	汎用次元数 汎用次元数範囲
<select-grank-construct> <select-grank-stmt> <select-grank-case-stmt>	SELECT GENERIC RANK 構文 SELECT GENERIC RANK 文 SELECT GENERIC RANK CASE 文
<select-gtype-construct> <select-gtype-stmt> <select-gtype-guard-stmt>	SELECT GENERIC TYPE 構文 SELECT GENERIC TYPE 文 汎用型捕捉文

x01. 総称副プログラム

モジュール副プログラムまたは内部副プログラムの SUBROUTINE 文または FUNCTION 文に書かれた GENERIC 接頭辞は、その副プログラムが総称副プログラムであることを指定する。その副プログラムは、名前が総称であり、また、総称名で特定される一つ以上の個別手続を定義する。個別手続の仮引数は、個々に型、種別及び次元数をもつ。それぞれの個別手続の引用仕様は明示的であるとする。

名前が既に総称名であるとき、新しい個別手続は既存の個別手続の集合に追加される。どの参照も曖昧でないことを保証するため、それらのどの二つの手続も、15.4.3.4.5 の規則を満たさなければならない。（訳注: Fortran 2023 の 15.4.3.4.5 は、同じ総称名をもつ任意の二つの個別手続が満たすべき規則を示している。任意の二つの個別手続について、引数の数が同じ場合、対応する仮引数の対のうち少なくとも一対は、区別可能でなければならない。区別可能とは、型、種別または次元数が異なる場合などである。）

制約: モジュール副プログラムが総称であるとき、その内部服プログラムは総称であってはならない。

コメント
これは、総称副プログラムの入れ子を避けるための単純化である。N の 2 乗個の個別手続（そのほとんどは使われないかもしれない）を生成することは、プロセッサにとって不必要な負担となる。入れ子の総称副プログラムの代わりに、横並びの総称副プログラムを使うことができる。それらがモジュール副プログラムの場合には、それらのうちのいくつかは PRIVATE にするのが良いだろう。

制約: 総称副プログラムは、選択戻り星印をもつことができない。

コメント
これは、廃止予定事項の利用の広がりを避けるためであり、技術的な理由ではない。

例 1

```
MODULE mod
CONTAINS
  GENERIC SUBROUTINE my_lift(x)
    TYPE(INTEGER, REAL) :: x
    ...
  END SUBROUTINE
END MODULE mod
```

例 2

```
PROGRAM main
  ...
CONTAINS
  GENERIC SUBROUTINE my_lift(x)
    TYPE(INTEGER, REAL) :: x
    ...
```

```
  END SUBROUTINE
END PROGRAM main
```

注釈: 内部副プログラム及びモジュール副プログラムだけが GENERIC 接頭辞をもつことができる。外部副プログラム、または、分離モジュール手続以外の引用仕様本体は、GENERIC 接頭辞をもつことはできない。

x01a. 総称分離モジュール副プログラム

分離モジュール手続引用仕様の SUBROUTINE 文または FUNCTION 文の GENERIC 接頭辞は、分離モジュール手続について、その名前が総称名であることと、GENERIC 接頭辞と MODULE 接頭辞の両方をもつモジュール副プログラムによって定義されることを宣言する。

コメント
この場合、“MODULE PROCEDURE <手続名>”の形の MODULE PROCEDURE 文は、次の理由により使用できない。

- 分離モジュール副プログラムが総称副プログラムであることを示していない。
- 同じ総称名をもつ総称副プログラムが複数あったとき、どれを実装しているかが曖昧になる。

x02. 総称引用使用宣言及び GENERIC 文

総称引用使用宣言の PROCEDURE 文中または GENERIC 文中に現れる個別手続並びは、総称名を指定するために次のように拡張される。

```
個別手続    is      手続名
            or      総称名
```

個別手続並び中に現れる総称名は、あたかも総称名によって特定される全ての個別手続がそこに書かれたかのように扱われる。

制約: 総称名が総称引用仕様宣言中の PROCEDURE 文の個別手続並び中に現れるとき、その総称指定は総称名であってはならない。総称名が GENERIC 文の個別手続並び中に現れるとき、その総称指定は総称名であってはならない。（訳注: 総称引用使用宣言の総称指定として、OPERATOR(演算子)、ASSIGNEMNT (=)、または、利用者定義入出力総称指定を使用することは許される。）

コメント
この制約は、総称名に対応する総称名を指定することを禁止している。それは、総称名の相互包含、再帰参照、あるいは評価順序の問題によって複雑な状況を引き起こす可能性がある。

無効な例

```
MODULE bad
  INTERFACE invalid
```

```
    PROCEDURE xyz
    PROCEDURE gsub      ! 上記の制約に違反
  END INTERFACE
CONTAINS
  SUBROUTINE xyz()
  END SUBROUTINE
  GENERIC SUBROUTINE gsub(a)
    TYPE(integer,real) :: a
    a = 999
  END SUBROUTINE
END MODULE
```

例

```
MODULE example
 INTERFACE OPERATOR(.myop.)
  PROCEDURE fun              ! 総称名 fun で指定されるすべての個別手続
  FUNCTION fen(a,b)          ! 外部手続 fen
   REAL, INTENT(IN) :: a, b
   REAL :: fen
  END FUNCTION fen
 END INTERFACE
CONTAINS
  GENERIC FUNCTION fun(a)
    REAL, INTENT(IN), RANK(0) :: a
    REAL, RANK(0) :: fun
    ...
  END FUNCTION fun
  GENERIC FUNCTION fun(a) RESULT(b)
    REAL, INTENT(IN), RANK(1:) :: a
    REAL, RANK(RANK(a)) :: b
    ...
  END FUNCTION fun
END MODULE
```

x03. 総称副プログラムの仮引数

汎用仮引数とは、型または種別について汎用的であるか、次元数について汎用的であるか、またはその両方である仮引数である。型または種別について汎用的な仮引数は、汎用型指定子によって宣言される。次元数について汎用的な仮引数は、汎用次元数指定子によって宣言される。

注釈: 汎用仮引数の型、種別または次元数に依存する型、種別または次元数をもつ言語要素は、汎用仮引数ではなく汎用依存言語要素であるという。この仕様については後述する。

汎用型指定子または汎用次元数指定子は、汎用型宣言文の中だけに書くことができる。汎用型宣言文は宣言構文であり、総称副プログラムの宣言部だけに書くことができる。

R8nn　汎用型宣言文 is　　汎用宣言型指定子 [[, 汎用属性指定子] ... ::] 汎用仮引数宣言並び
　　or　　宣言型指定子 , 汎用属性指定子並び :: 汎用仮引数宣言並び

C8nn　汎用型宣言文が汎用宣言型指定子をもたないとき、その汎用属性指定子並びは、汎用次元数指定子を含まなければならない。

R8nn　汎用仮引数宣言　　is　　仮引数名 [(配列形状指定)] [* 文字長]

C8nn　汎用仮引数は、共配列であってはならない。

コメント
共次元については、requirements、use cases 及び specifications のどこにも言及されていないが、複雑さを増加させると思われるので、今のところは禁止としておく。

C8nn　汎用仮引数は、省略可能でない仮データ実体でなければならない。
（訳注: つまり、仮手続は汎用仮引数になれない。）

コメント
省略可能でないことは必須である。そうでなければ、生成される個別手続が曖昧になってしまう。

Formal specifications に追加すべき事項

sNN　汎用仮引数は共配列になれない。

理由: 単純化のためである。将来許すかもしれない。共次元数について汎用な仮引数も許すかもしれない。

sNN　汎用仮引数は仮データ実体でなければならず、手続であってはならない。

理由: これも単純化のためである。構文的には、汎用宣言型指定子及び汎用次元数指定子を、総称副プログラム中の引用仕様本体に書くことを認める必要が生じる。

ここで、仮手続の問題を考える。

三つの選択肢がある。最も厳しいもの（最も単純化できるもの）から順に次の通りである。

(1) 制約: 総称副プログラムは、仮手続をもつことができない。
(2) 制約: 総称副プログラムの仮手続は、明示的引用仕様をもたなければならず、かつ、汎用または汎用依存であってはならない。

(3) 制約: 総称副プログラムの仮手続は、明示的引用仕様をもたなければならず、かつ、汎用であってはならないが、汎用依存であってもよい。

コメント（Malcolm）

(a) 明示的引用仕様を必須とするのは単純化のためだが、安全性も向上させている。
(b) 既に述べた通り、我々は手続が汎用仮引数になれないという制約を設けた。
(c) 単純化(1)は厳しすぎると思われる（汎用性とは無関係なコールバックには問題はないはずである）。従って、単純化(2)が検討すべき最低限のものとなる。しかしそれだと、引数や結果の型が仮引数に依存する関数は除外されることになり、機能性が損なわれる。

提案: 今のところ、(3)で行きましょう。

x04. 種別について汎用的な仕様に関する構文

R8nn	種別汎用型指定子	is	汎用組込み型指定子
		or	汎用派生型指定子
R8nn	汎用組込み型指定子	is	非文字組込み型名 ([KIND =] 整定数式)
		or	CHARACTER (汎用文字型パラメタ)
R8nn	非文字組込み型名	is	REAL | INTEGER | LOGICAL | COMPLEX
R8nn	汎用文字型パラメタ	is	汎用文字長 , [KIND =] 整定数式
		or	LEN = 汎用文字長 , KIND = 整定数式
		or	KIND = 整定数式 , LEN = 汎用文字長
R8nn	汎用文字長	is	* | :

C8nn 汎用組込み型指定子中の整定数式は、一次元配列でなければならない。

コメント

種別汎用型指定子は、種別型パラメタが配列であることを除けば、見かけは通常の型指定子と同じである。

例 x04-1

```
GENERIC SUBROUTINE gensub(x,y)
  INTEGER([int8,int16,int32]),INTENT(INOUT) :: x
  CHARACTER(*,KIND=[ascii,iso_10646]) :: y
```

R8nn	汎用派生型指定子	is	型名 (汎用型パラメタ指定並び)
R8nn	汎用型パラメタ指定	is	[キーワード =] 汎用型パラメタ値
R8nn	汎用型パラメタ値	is	整定数式 | * | :

C8nn　汎用型パラメタ値は、長さ型パラメタのとき、* または : でなければならず、そうでないとき、一次元配列である整定数式でなければならない。

C8nn　汎用派生型指定子には、少なくとも一つの種別型パラメタを指定しなければならない。

例 x04-2

```
TYPE T(k1,k2,n)
  INTEGER,KIND :: k1,k2
  INTEGER,LEN :: n
  REAL(k1) value(k2,n)
END TYPE
GENERIC SUBROUTINE gensub2(x)
  TYPE(t([kind(0.0),kind(0d0)],k2=[1,2,4,8],n=*)),INTENT(INOUT) :: x
```

この例は、k1 として 2 つの値をもち、それとは独立に k2 として 4 つの値をもつので、次の 8 つの個別手続をカバーしている。

```
TYPE(t(k1=kind(0.0),k2=1,n=*)
TYPE(t(k1=kind(0.0),k2=2,n=*)
TYPE(t(k1=kind(0.0),k2=4,n=*)
TYPE(t(k1=kind(0.0),k2=8,n=*)
TYPE(t(k1=kind(0d0),k2=1,n=*)
TYPE(t(k1=kind(0d0),k2=2,n=*)
TYPE(t(k1=kind(0d0),k2=4,n=*)
TYPE(t(k1=kind(0d0),k2=8,n=*)
```

x04a. 汎用組込み型指定子の拡張（オプション）

組込み型に限り、「すべての可能な種別値」を意味する特別な構文をもつ。

R8nn generic 組込み型指定子	is	非文字組込み型名 ([KIND =] generic 種別)
	or	CHARACTER (generic 文字型パラメタ)
R8nn 非文字組込み型名	is	REAL | INTEGER | LOGICAL | COMPLEX
R8nn generic 文字型パラメタ	is	generic 文字長 , [KIND =] generic 種別
	or	LEN = generic 文字長 , KIND = generic 種別
	or	KIND = generic 種別 , LEN = generic 文字長
R8nn generic 種別	is	整定数式
	or	*

コメント

REAL(*) は REAL(real_kinds)と同じ効果をもち、INTEGER(*) は INTEGER(integer_kinds)と同じ効果をもつので（他も同様）、必須ではない。しかし、おそらく * の方が見栄えが良いだろう。また、real_kinds などを得るために組込みモジュール ISO_FORTRAN_ENV を使う必要がない。

x04b. 種別値の重複を許す（オプション 2）

種別型パラメタを表す整定数式（一次元配列）の中で、値の重複を許すか（重複した値は無視）？

Yes の場合、normative text:

> “種別汎用型指定子の中では、種別値は重複してもよい。重複があるとき、あたかも一つだけ書いたかのように扱われる。”

No:

> C8nn　種別汎用型指定子中に指定された種別値は、互いに異なっていなければならない。

例えば、利用者はおそらく次のように書きたいだろう。

```
TYPE(REAL([selected_real_kind(3),selected_real_kind(2)])) x
```

16 ビット実数をサポートしていない処理系、または、16 ビット実数を一種別しかもたない処理系では、値は重複する。IEEE の 16 ビットと bfloat16 の両方をもつ処理系では、別々の値となる。

x05. 型について汎用的な仕様に関する構文

R8nn	汎用宣言型指定子	is	TYPE (汎用型指定子並び)
		or	CLASS (汎用型指定子並び)
		or	汎用組込み型指定子
R8nn	汎用型指定子	is	組込み型指定子
		or	派生型指定子
		or	enumeration 型指定子
		or	種別汎用型指定子

C8nn　汎用宣言型指定子のキーワードが CLASS であるとき、汎用指定子はそれぞれ一つの拡張可能型を識別しなければならない。

C8nn　汎用型指定子並びは、種別汎用型指定子を含んでいないとき、二つ以上の項目をもたなければならない。

C8nn　汎用型指定子は、すべての長さ型パラメタに、引継ぎ型パラメタまたは無指定型パラメタを指定しなければならない。

コメント
長さ型パラメタは総称の解決に関与しないので、この単純化は主に利用者のケガを防ぐためである。（訳注: この単純化により、長さ型パラメタに整数値を指定することができなくなる。これは機能性を損なう恐れがある。）

例 x05-1

```
GENERIC FUNCTION plus(a,b) RESULT(r)
  TYPE(integer,real,complex),INTENT(IN) :: a
  TYPEOF(a),INTENT(IN) :: b
  TYPEOF(a) :: r
  r = a + b
END FUNCTION
```

これは、次の三つの個別手続を生成する。

```
integer function(integer a,b)
real function(real a,b)
complex function(complex a,b)
```

例 x05-2

```
TYPE t1
  ...
END TYPE
TYPE t2
  ...
END TYPE
GENERIC SUBROUTINE process(x)
  CLASS(t1,t2),INTENT(IN) :: x
  ...
END SUBROUTINE
```

これは、二つの個別手続を生成する。一つがもつ引数は CLASS(t1)であり、もう一つがもつ引数は CLASS(t2)である。

x05a. 型と種別の重複を許すべきか（オプション 3）

型と種別が重複している例

```
USE ISO_FORTRAN_ENV
TYPE(REAL(REAL64), DOUBLE PRECISION) :: x
TYPE(INTEGER, INTEGER(INT32)) :: y
```

ある処理系では、それぞれの汎用型宣言文で、1 番目と 2 番目の型指定子は同じ型及び型パラメタを指定しているが、別の処理系ではそうではない。

Yes の場合、normative text:

“汎用型宣言文中の汎用型指定子には、同じ型及び型パラメタを指定することが許される。冗長な指定が無視されても、仮引数は汎用仮引数である。”

No の場合、制約:

C8nn 汎用型宣言文中の汎用型指定子には、同じ文中の別の汎用型宣言子と同じ型及び型パラメタを指定してはならない。

x06. 次元数について汎用的な仕様

次元数について汎用的な仮引数は、汎用次元数指定子を使って宣言する。型または種別が汎用的でなくても、汎用次元数指定子をもつことで型宣言文は汎用型宣言文となる。

R8nn	汎用次元数指定子	is	RANK (汎用次元数並び)
R8nn	汎用次元数	is or or	スカラ整定数式 汎用次元数範囲 *
R8nn	汎用次元数範囲	is	スカラ整定数式 : スカラ整定数式

C8nn 汎用次元数並び中のスカラ整定数式は、負の値であってはならない。

C8nn 汎用次元数並びは、* または汎用次元数範囲を含んでいないとき、二つ以上のスカラ整定数式を含んでいなければならない。

オプション
RANK(*) は通常の場合大きさ引継ぎを意味するので、避けた方が良いかもしれない。

オプションここまで

汎用次元数としての * は、処理系が対応できるすべての次元数を指定する。

（訳注: 原案 (N2217) では、汎用次元数として * を許していない。代わりに、汎用次元数範囲のスカラ整定数式をどちらも省略することができ、両方省略した場合の : は、この * と同じ意味をもつ。）

C8nn 汎用次元数並びには、同じ次元数を一度しか指定することができない。

コメント
これについては、重複を許しても便利だとは思われない。

オプション 4
もし次元数の重複を許すなら、normative text は次の通り:

“汎用次元数並びによって重複した値を指定することが許されるが、重複した値は無視される。結果として一つの次元数だけが有効であったとしても、その仮引数は汎用的であるとする。”

オプション 4 ここまで

例 x06

```
GENERIC SUBROUTINE lift(x,y)
  TYPE(INTEGER(int32,int64), REAL), RANK(1:2), ALLOCATABLE :: x, y
  TYPEOF(x),RANKOF(y),ALLOCATABLE :: z
  ...
END SUBROUTINE
```

このサブルーチンは総称名 lift をもつ 36 の個別手続を定義する。変数 x, y, z は割付け可能であり、他の性質はそれぞれ次の通りである。

int32, rank 1	int32, rank 1	int32, rank 1
int64, rank 1	int32, rank 1	int64, rank 1
real, rank 1	int32, rank 1	real, rank 1
int32, rank 2	int32, rank 1	int32, rank 1
int64, rank 2	int32, rank 1	int64, rank 1
real, rank 2	int32, rank 1	real, rank 1
int32, rank 1	int32, rank 2	int32, rank 2
...		
real, rank 2	real, rank 2	real, rank 2

（訳注: つまり、次のように書いた場合と同じ意味になる。

```
TYPE(INTEGER(int32,int64), REAL), RANK(1:2), ALLOCATABLE :: x
TYPE(INTEGER(int32,int64), REAL), RANK(1:2), ALLOCATABLE :: y
TYPEOF(x),RANKOF(y),ALLOCATABLE :: z
```

原案（N2217）では、x と y を同じ行に書いた場合には、次の例と同じ意味を持つ 6 つの個別手続を定義するとしていた。）

例 x06-2

```
GENERIC SUBROUTINE lift(x,y)
  TYPE(INTEGER(int32,int64), REAL), RANK(1:2), ALLOCATABLE :: x
  TYPEOF(x), RANKOF(x), ALLOCATABLE :: y
  TYPEOF(x),RANKOF(y),ALLOCATABLE :: z
  ...
END SUBROUTINE
```

この副プログラムは、総称名 lift をもつ 6 つの個別手続を定義する。

x07. 最大次元数の表現

汎用次元数を指定するとき、最小値よりも大きなすべての有効な次元数（例えば、次元数 2 以上のすべての配列）を指定できることが望ましい場合がある。二つの可能性が考えられる。

(1) 処理系が用意する最大の次元数を超えた値を指定することを許す。ただしそれは無視される。
(2) 処理系が用意する最大の次元数を表現するための構文または名前付き定数を提供する。構文として汎用次元数範囲の中に “*” があってもよいし、名前付き定数として MAX_RANK があってもよい。（訳註: この * は x06 で定義された * と意味が一貫していない。）

提案: 組込みサブルーチン ISO_FORTRAN_ENV に新しい名前付き定数 MAX_RANK を追加する。その値は、非共配列に対して処理系が用意する最大の次元数

（訳註: 原案（N2217）では、汎用次元数範囲中の上限のスカラ整定数式を省略することで、処理系が用意する最大の次元数を表現する。例えば、「次元数 2 以上のすべての配列」の表現は “2 : ” である。）

x08. 汎用依存言語要素

局所言語要素（仮引数及び関数結果を含む）は、事前に宣言された汎用仮引数の型、種別または次元数をもつことを宣言することが許される。そのような言語要素は、汎用依存言語要素 (generic-dependent entity) と呼ばれ、それが依存する汎用仮引数と同じ型、種別及び／または次元数をもつことになる。これは、TYPEOF 型指定子及び CLASSOF 型指定子、汎用仮引数を引数とする KIND 関数、並びに、RANKOF 節（新規）を使って実現することができる。

（拡張）R802　属性指定子　...
　　or　RANKOF (仮引数名)

C8nn　RANKOF 節中の仮引数名は、汎用仮引数の名前でなければならない。

C8nn　属性指定子並び中に RANKOF 節が現れるとき、RANK 節または DIMENSION 節は現れてはならない。

C8nn　言語要素の次元数は、RANKOF 節を介さない限り、汎用仮引数の次元数に依存してはならない。

例

```
GENERIC SUBROUTINE lift(x,y)
  TYPE (INTEGER, REAL), ALLOCATABLE, RANK(1:2) :: x
  TYPEOF(x), RANKOF(x) :: y
  ...
END SUBROUTINE
```

x09. SELECT GENERIC RANK 構文

総称副プログラム内の SELECT GENERIC RANK 構文は、生成される個別副プログラム毎に、高々一つのブロックを選択する。

R1150a SELECT GENERIC RANK 構文 is SELECT GENERIC RANK 文
[SELECT GENERIC RANK CASE 文ブロック] ...
END SELECT GENERIC RANK 文

R1151a SELECT GENERIC RANK 文 is
[SELECT 構文名 :] SELECT GENERIC RANK (選択子)

C1155a SELECT GENERIC RANK 文中の選択子は、次元数について汎用的な仮引数でなければならない。

R1152a SELECT GENERIC RANK CASE 文
is RANK (汎用次元数並び) [SELECT 構文名]
or RANK DEFAULT [SELECT 構文名]

C1152a SELECT GENERIC RANK CASE 文中の汎用次元数は * であってはならない。

汎用次元数並びは x07 で定義されている。

値の重複については同じように扱われる（オプション 4 参照）。

C1157a SELECT GENERIC RANK 構文内では、二つ以上の SELECT GENERIC CASE 文中に同じ値を指定してはならない。

C1158a SELECT GENERIC RANK 構文内では、RANK DEFAULT 文は高々一つしか現れてはならない。

R1153a END SELECT GENERIC RANK 文 is
END SELECT [SELECT 構文名]

SELECT 構文名に関する規則は、SELECT CASE 構文におけるそれと同様である。

SELECT GENERIC RANK 文:

SELECT GENERIC RANK (x)

をもつ SELECT GENERIC RANK 構文の実行は、SELECT CASE 文:

SELECT CASE (RANK (x))

をもつ SELECT CASE 文の実行と同様である。

コメント

それぞれの個別副プログラムには、高々一つの実行コードが含まれることになる。SELECT CASE 構文とは違い、SELECT GENERIC RANK 構文では、ブロックの選択と分岐のための実行時オーバーヘッドは発生しないであろう。

例

```
GENERIC FUNCTION fun(x) RESULT(y)
  TYPE(type1), RANK(0:7) :: x
  TYPEOF(x), RANK(RANK(x)) :: y

  SELECT GENERIC RANK (y)
  RANK (0)
    !! code if y is a scalar
  RANK (1:3)
    !! code if y is an array of 1 to 3 dimensions
  RANK DEFAULT
    !! code if y is an array of 4 to 7 dimensions
  END SELECT
END FUNCTION fun
```

x10. SELECT GENERIC TYPE 構文

総称副プログラム内の SELECT GENERIC TYPE 構文は、生成される個別副プログラム毎に、高々一つのブロックを選択する。

コメント

この選択は、汎用型言語要素または汎用種別言語要素の宣言時の型を基に行われる。実行時の型ではない。

R1154a SELECT GENERIC TYPE 構文 is SELECT GENERIC TYPE 文
[汎用型捕捉文
ブロック] ...
END SELECT GENERIC TYPE 文

R1155a SELECT GENERIC TYPE 文 is
[SELECT 構文名 :] SELECT GENERIC TYPE (選択子)

C1163a SELECT GENERIC TYPE 文中の選択子は、型または種別について汎用的な仮引数でなければならない。

R1156a 汎用型捕捉文 is TYPE IS (型指定子) [SELECT 構文名]
or TYPE DEFAULT [SELECT 構文名]

C1165a型指定子は組込み型指定子または派生型指定子でなければならず、かつ、型指定子の長さ型パラメタは引継ぎでなければならない。

C1158aSELECT GENERIC TYPE 構文内では、二つ以上の TYPE IS 文中に、同じ型かつ同じ種別型パラメタを指定してはならない。

C1169aSELECT GENERIC TYPE 構文内では、TYPE DEFAULT 文は高々一つしか現れてはならない。（訳註: 原文には 2 箇所誤りがあって修正した。<generic-type-construct> 及び CLASS DEFAULT）

R1157aEND SELECT GENERIC TYPE 文 is
END SELECT [SELECT 構文名]

実行

SELECT GENERIC TYPE 構文は、実行される 1 ブロックを選択する。選択子が多相的であるか否かに関わらず、実行されるブロックは、選択子の宣言時の型と種別によって選択される。それが一つの TYPE IS 文の型指定子と一致したとき、その文に続くブロックが選択される。そうでないとき、TYPE DEFAULT 文があればその文に続くブロックが選択され、なければどのブロックも選択されない。

コメント

ブロックは翻訳時に選択される。SELECT GENERIC TYPE 構文の実行は、選択されたブロック（あれば）の実行である。

例

```
GENERIC FUNCTION fun(x) RESULT(y)
  TYPE(type1,type2) :: x, y
  !! code if x is type1 or type2
  SELECT GENERIC TYPE (x)
  TYPE IS (type1)
    !! code if x is type1
  TYPE IS (type2)
   !! code if x is type2
  END SELECT
END FUNCTION fun
```

3. さらに他の例

例 1

```
MODULE example
  INTERFACE OPERATOR(.myop.)
    PROCEDURE s ! All of the specific procedures of s.
  END INTERFACE
CONTAINS
```

```
    GENERIC FUNCTION s(a,b) RESULT(c)
      TYPE(REAL,COMPLEX), INTENT(IN), RANK(*) :: a
      TYPEOF(a),RANKOF(a), INTENT(IN) :: b
      TYPEOF(a), RANKOF(a) :: c, temp
      ...
      SELECT GENERIC TYPE (a)
      TYPE IS (REAL)
        temp = temp * (1-b)
      TYPE IS (COMPLEX)
        ! Just this once, we want the conjugate.
        temp = temp * (1-CONJG(b))
      END SELECT
      ...
      c = temp
    END FUNCTION
  END MODULE
```

例 2

```
PROGRAM main
  INTEGER:: n = 5
  WRITE(*,*) factorial(n)
CONTAINS
  GENERIC RECURSIVE FUNCTION factorial(n) RESULT (res)
    USE ISO_FORTRAN_ENV
    INTEGER (int_kinds) :: n
    TYPEOF(n) :: res
    IF (n >1) THEN
     res = n*factorial(n-1)
    ELSE  IF (n==1) THEN
     res = 1
    ELSE
     res =0
    END IF
  END FUNCTION factorial
END PROGRAM main
```