

**N1207**

**International Standards Organisation**

**Parameterized Derived Types**

**in**

**Fortran**

Technical Report defining extension to  
ISO/IEC 1539-1 : 1996

{Produced 16/7/96 }

THIS PAGE TO BE REPLACED BY ISO CS

<b>1 : GENERAL</b>	<b>5</b>
1.1 Scope	5
1.2 Normative References	5
<b>2 : RATIONALE</b>	<b>2</b>
<b>3 : REQUIREMENTS</b>	<b>3</b>
<b>3.1 Description of parameterized derived type enhancements</b>	<b>3</b>
3.1.1 The Type Definition	4
3.1.2 The Type Definition	5
3.1.3 Object declaration	6
3.1.4 The form of the Constructor	7
3.1.5 Type parameter value inquiry	9
3.1.6 Intrinsic assignment	10
3.1.7 Argument association and overload rules	10
3.1.8 Visibility and Scoping rules	10
<b>4 REQUIRED EDITORIAL CHANGES TO ISO/IEC 1539-1 : 1996</b>	<b>12</b>
4.1.1 Edits to implement parameterized derived types	13
4.1.2 Edits to implement parameter value inquiry	16
4.1.3 Edits to implement structure constructor	17
4.1.4 Additional edits to implement constructors as generic functions	17
4.1.5 Declaration of objects	19

## **Foreword**

[This page to be provided by ISO CS]

## Introduction

This technical report defines a proposed extension to the data-typing facilities of the programming language Fortran. The current Fortran language is defined by the international standard ISO/IEC 1539-1 : 1996. This technical report has been prepared by ISO/IEC JTC1/SC22/WG5, the technical working group for the Fortran language. The language extension defined by this technical report is intended to be incorporated in the next revision of the Fortran language without change except where experience in implementation and usage indicates that changes are essential. Such changes will only be made where serious errors in the definition or difficulties in integration with other new facilities are encountered.

This extension is being defined by means of a technical report in the first instance to allow early publication of the proposed definition. This is to encourage early implementation of important extended functionalities in a consistent manner and will allow extensive testing of the design of the extended functionality prior to its incorporation into the language by way of the revision of the international standard.

# Information technology - Programming Languages - Fortran

## Technical Report: Parameterized Derived Types

### 1 : General

#### 1.1 Scope

This technical report defines a proposed extension to the data-typing facilities of the programming language Fortran. The current Fortran language is defined by the international standard ISO/IEC 1539-1 : 1996. The enhancements defined in this technical report extends the capability of parameterization defined for intrinsic types to derived types.

Section 2 of this technical report contains a general informal but precise description of the proposed extended functionalities. This is followed by detailed editorial changes which if applied to the current international standard would implement the revised language definitions.

#### 1.2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this technical report. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this technical report are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 1539-1 : 1996 *Information technology - Programming Languages - Fortran*

## 2 : Rationale

Parameterized derived types are required for two main reasons. Firstly, there are many circumstances where a derived type is required to work together with intrinsic types where the ability to parameterize the kind of the latter and not the former causes very considerable problems. In one case different versions of the program can be selected by the use of the parameter but to enable the derived type to properly interwork a different type with a different name must be used. This results in very clumsy and inflexible programs and a significant program maintenance overhead, substantially defeating the object of the kind parameterization. Secondly, there are a large number of types where there is a need to manipulate objects where the only difference between various entities is in the size of some internal component. For example, there are entities like vectors that may differ in the dimensionality of the space they span and therefore in the number of reals that are involved in their representation, or in matrices that differ in their order. These are very like the intrinsic character data type where data objects may differ in the number of characters in the string and where this is specified by a length parameter on the type. This is clearly preferable to having multiple separate types which differ only in such a size determining property. Both these requirements are met by the addition of parameterized derived types to the language.

## 3 : Requirements

The following subsections contain a general description of the extensions required to the syntax and semantics of the current Fortran language to provide for user defined parameterized derived types.

### 3.1 Description of parameterized derived type enhancements

There are seven main areas of language design where an extension such as this impacts the existing language and where syntax and semantics must be defined. These are:

- the definition of the type,
- declaration of objects of such a type,
- constructing a value of such a type,
- inquiring as to the value of a type parameter for an existing object of such a type,
- intrinsic assignment for objects of such a type,
- argument association and overload resolution, and
- the visibility and scoping rules.

Syntactic forms and semantic rules exist covering the use of parameterized intrinsic types in all but the first of these areas; for obvious reasons there is no type definition for an intrinsic type.

In this section the technical nature of the proposal in each of the above areas is covered with sufficient detail to indicate the essential nature of the proposed syntax and semantics. This is done informally with the approach illustrated by example rather than with detailed syntactic and semantic rules. The formal rules will be defined in subsequent sections in the form of proposed edits to the current international standard for the programming language Fortran which would implement the proposed extensions.

All parameters for intrinsic types are quantities of type default integer. This technical report proposes that parameters for derived types be similarly restricted at this time. However, the detailed form of the extension defined in this technical report is such that parameters of other types could be added by further extension if that proves to be desirable.

The intrinsic types have parameters of two quite different natures. There are the static parameters that determine the nature of the machine representation. These are all characterised for the intrinsic types by the same parameter name, `KIND`. This is used both for the keyword in the *type-spec* and as the generic name of the parameter-value inquiry function for such a parameter. `KIND` parameters can be used to resolve overloads.

The other parameter variety, where the value is not necessarily static, only applies intrinsically for the character type. Here the parameter, `LEN`, determines the length or the number of characters in the datum. As for `KIND`, the name `LEN` is also both the parameter keyword name and the generic name of the inquiry function used to find the value of the parameter for an appropriate data object.



















## **4 Required editorial changes to ISO/IEC 1539-1 : 1996**

The following subsections contain the editorial changes to ISO/IEC 1539-1 : 1996 required to include these extensions in a revised definition of the international standard for the Fortran language. Note, where new syntax rules are inserted they are numbered with a decimal addition to the rule number that precedes them. In the actual document these will have to be properly numbered in the revised sequence.

Comments about each edit to the standard appear within braces { }.

N.B. In this draft the edits refer to X3J3/96-007R1, April 22, 1996, 8:35 a.m.

















After "type" add "and type parameters"

7.1.4.2 [92/17]

After the second "The type" add "and type parameters."

7.1.6.1 [93/25+ &94/17+]

Add new list item and renumber the next list item as (f)

(e) a derived-type static type parameter inquiry expression, or

7.1.6.2 [95/36]

Add new list item and renumber the next list item as (f)

(e) a derived-type type parameter inquiry expression"

7.1.7 [97/40]

Add paragraph

The appearance of a structure constructor requires the evaluation of the component expressions and may require the evaluation of type parameter expressions. The type of an expression in which a structure constructor appears does not affect, and is not effected by, the evaluation of such expressions, except that evaluation of the static type parameters may affect the resolution of a generic reference to a defined operation or function and hence may affect the expression type.

7.5.1.2 [107/39]

Replace "type," by "type and the same type parameter values,"

7.5.1.2 [108/8]

Replace "type as" by "type and the same type parameter values as"

11.3.2 [187/31+]

Add sentence

If a derived type type parameter name is renamed, the local name is used for the type parameter keyword name used when specifying actual type parameter values.

12.2.1.1 [192/17]

Replace "or character length" by " character length, or nonstatic type parameter"

12.3.1.1 [193/18]

Replace "that" by "that assumes the value for a nonstatic derived type parameter or that"

12.3.1.1 [193/22]

Add additional item to list and renumber list

(d) A result with a nonconstant type parameter value (derived type functions only)

12.4.1.1 [200/5]

Add sentence after "dummy argument."



The value of a type parameter of an actual argument of derived type must agree with the corresponding value for the dummy argument.

14.1.2 [275/38]

Replace ", in" by " and type parameters, in"

14.1.2.3 [277/19]

[277/22]

[277/23]

[277/27]

[277/31]

Replace "kind" by "static"