INTERNATIONAL STANDARD

ISO/IEC 1539-1

First edition
1997-12-15

Information technology — Programming languages — Fortran —

Part 1: Base language

Technologies de l'information — Langages de programmation — Fortran —

Partie 1: Langage de base

Reference number
ISO/IEC 1539-1:1997(E)

Contents

Tables

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of techical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication of an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO/IEC 1539-1 was prepared by Joint Technical Committee ISO/IEC/JTC1, *Information technology,* Subcommittee SC22, *Programming languages, their environments and system software interfaces.*

This edition cancels and replaces ISO/IEC 1539:1991, which has been technically revised.

ISO/IEC 1539 consists of the following parts, under the general title *Information technology — Programming languages — Fortran*:

— *Part 1: Base language*

— *Part 2: Varying length character strings*

Annexes A to D of this part of ISO/IEC 1539 are for information only.

Introduction

Standard programming language Fortran

This part of the international standard comprises the specification of the base Fortran language. With the limitations noted in 1.5.1, and the deletions described in Annex B, the syntax and semantics of Fortran 90 are contained entirely within Fortran 95. Therefore, any standard-conforming Fortran 90 program not containing deleted features or affected by such limitations is a standard conforming Fortran 95 program. New features of Fortran 95 can be compatibly incorporated into such Fortran 90 programs, with any exceptions indicated in the text of this part of the standard.

Fortran 95 continues the evolutionary model introduced in Fortran 90 by deleting several of the features marked as obsolescent in Fortran 90 and identifying a few newly-obsolescent features (Annex B).

Fortran 95 is a relatively minor evolution of standard Fortran, with the emphasis in this revision being upon correcting defects in the Fortran 90 standard, including providing interpretation to a number of questions that have arisen concerning Fortran 90 semantics and syntax (e.g., whether blanks are permitted within edit descriptors in free source form). In addition to such corrections and clarifications, Fortran 95 contains several extensions to Fortran 90; there are three major extensions:

(1) The FORALL statement and construct
(2) PURE and ELEMENTAL procedures
(3) Pointer initialization and structure default initialization

FORALL

The Fortran 90 array constructor and SPREAD and RESHAPE intrinsic functions are powerful tools for element-by-element construction of an array value. Their use in combination, which is required for many array values, can be awkward. Fortran 95 therefore provides a simple and efficient alternative: the FORALL statement allows array elements, array sections, character substrings, or pointer targets to be explicitly specified as a function of the element subscripts. The form of the FORALL statement is very much like a functionally equivalent set of nested DO loops for computing and assigning the elements of an array, except that conceptually all elements are computed simultaneously and then assigned simultaneously. An added benefit of FORALL is that it simplifies conversion from sequential DO loops to parallel array operations. A FORALL construct allows several such array assignments to share the same element subscript control. This control includes masking in a manner similar to the masking facilities of WHERE, the main difference between WHERE and FORALL being that FORALL makes use of element subscripts whereas WHERE is whole array oriented.

PURE

As has always been the case in Fortran, Fortran 95 functions may have side effects (e.g., change the value of an argument or a global variable). Side effects cause problems in parallel processing, however, and because parallel processing has become an important high performance technology, Fortran 95 makes it possible to specify a function to be side effect free. Such a function is called "pure" and is declared with the keyword PURE in the function statement. A restricted form of PURE functions may be called elementally; such ELEMENTAL functions are especially important to high performance parallel processing. An added advantage of pure functions is that it is reasonable to allow them in specification expressions; this provides a significant amount of functionality, with very little cost, and therefore this capability has also been included in Fortran 95.

Initialization

In Fortran 90 there was no way to define the initial pointer association status — a pointer has to be explicitly nullified, allocated, or associated with a target during execution before it can be tested by the ASSOCIATED intrinsic function. This limits the usefulness of pointers, especially the use of pointers as derived-type components. Fortran 95 therefore solves this problem by providing (a) a NULL intrinsic function that may be used to nullify a pointer and (b) a means to specify default initial values for derived-type components. In the latter case the specification of initial values is part of the derived-type definition, and objects declared of this type automatically have all their components so initialized.

Organization of this part of ISO/IEC 1539

This part of ISO/IEC 1539 is organized in 14 sections, dealing with 7 conceptual areas. These 7 areas, and the sections in which they are treated, are :

High/low level concepts	Sections 1, 2, 3
Data concepts	Sections 4, 5, 6
Computations	Sections 7, 13
Execution control	Section 8
Input/output	Sections 9, 10
Program units	Sections 11, 12
Scoping and association rules	Section 14

High/low level concepts

Section 2 (Fortran terms and concepts) contains many of the high level concepts of Fortran. This includes the concept of a program and the relationships among its major parts. Also included are the syntax of program units, the rules for statement ordering, and the definitions of many of the fundamental terms used throughout this part of ISO/IEC 1539.

Section 3 (Characters, lexical tokens, and source form) describes the low level elements of Fortran, such as the character set and the allowable forms for source programs. It also contains the rules for constructing literal constants and names for Fortran entities, and lists all of the Fortran operators.

Data concepts

The array operations and data structures provide a rich set of data concepts in Fortran. The main concepts are those of data type, data object, and the use of data objects, which are described in Sections 4, 5, and 6, respectively.

Section 4 (Intrinsic and derived data types) describes the distinction between a data type and a data object, and then focuses on data type. It defines a data type as a set of data values, corresponding forms (constants) for representing these values, and operations on these values. The concept of an intrinsic data type is introduced, and the properties of Fortran's intrinsic types (integer, real, complex, logical, and character) are described. Note that only type concepts are described here, and not the declaration and properties of data objects.

Section 4 also introduces the concept of derived (user-defined) data types, which are compound types whose components ultimately resolve into intrinsic types. The details of defining a derived type are given (note that this has no counterpart with intrinsic types; intrinsic types are predefined and therefore need not - indeed cannot - be redefined by the programmer). As with intrinsic types, this section deals only with type properties, and not with the declaration of data objects of derived type.